

# ConvexOS Man Pages for System Managers

*First Edition*



CONVEX

CONVEX COMPUTER CORPORATION



*Peter Fink*

---

# ConvexOS Man Pages for System Managers



---

Order No. DSW-333

First Edition  
April 1991

CONVEX Press  
Richardson, Texas USA

---

## **ConvexOS Man Pages for System Managers**

Order No. DSW-333

©1991 CONVEX Computer Corporation.  
All rights reserved.

This document is copyrighted. This document may not, in whole or part, be copied, duplicated, reproduced, translated, electronically stored, or reduced to machine readable form without prior written consent from CONVEX Computer Corporation.

Although the material contained herein has been carefully reviewed, CONVEX Computer Corporation does not warrant it to be free of errors or omissions. CONVEX reserves the right to make corrections, updates, revisions or changes to the information contained herein. CONVEX does not warrant the material described herein to be free of patent infringement.

UNLESS PROVIDED OTHERWISE IN WRITING WITH CONVEX COMPUTER CORPORATION (CONVEX), THE PROGRAM DESCRIBED HEREIN IS PROVIDED AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES. THE ABOVE EXCLUSION MAY NOT BE APPLICABLE TO ALL PURCHASERS BECAUSE WARRANTY RIGHTS CAN VARY FROM STATE TO STATE. IN NO EVENT WILL CONVEX BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING ANY LOST PROFITS OR LOST SAVINGS, ARISING OUT OF THE USE OR INABILITY TO USE THIS PROGRAM. CONVEX WILL NOT BE LIABLE EVEN IF IT HAS BEEN NOTIFIED OF THE POSSIBILITY OF SUCH DAMAGE BY THE PURCHASER OF ANY THIRD PARTY.

CONVEX and the CONVEX logo ("C") are registered trademarks of CONVEX Computer Corporation.

UNIX is a trademark of AT&T Bell Laboratories.

Printed in the United States of America

---

## Revision Information for

### ConvexOS Man Pages for System Managers

---

Edition	Document No.	Description
First Edition	710-015930-000	April, 1991. Initial Release.

---

---

---

# Contents

---

<b>Using ConvexOS man pages.....</b>	<b>V</b>
What are man pages?	
Where to find other ConvexOS man pages	
How to use ConvexOS man pages	
Technical assistance	

---

## **Section 8—system management**

intro(8)  
ac(8)  
accounting(8)  
acctconv(8)  
accton(8)  
adbccu(8)  
ansidaemon(8)  
arp(8c)  
atrun(8)  
avail(8)  
catinfo(8)  
catman(8)  
chall(8)  
charge(8)  
chown(8)  
clri(8)  
comsat(8c)  
connecttime(8)  
cpuconf(8)  
crashdump(8)  
crashread(8)  
ctar(8)  
dcheck(8)  
diskuse(8)  
dump(8)  
dumpsfs(8)  
edactwho(8)  
edquota(8)  
faillogon(8)  
faillogpr(8)  
fastboot(8)  
fasthalt(8)  
fingerd(8c)  
fsck(8)  
fsirand(8)  
fstat(8)

ftpd(8c)  
genrest(8)  
getname(8)  
getst(8)  
gettable(8c)  
getty(8)  
halt(8)  
hconfig(8)  
hhstart(8)  
hhstop(8)  
hsend(8)  
htable(8c)  
hyroute(8c)  
hystat(8c)  
icheck(8)  
ifconfig(8c)  
inetd(8c)  
init(8)  
inline(8)  
installsw(8)  
lim(8)  
lockd(8c)  
lpc(8)  
lpd(8)  
lpf(8)  
lprewind(8)  
makedev(8)  
makekey(8)  
makewhatis(8)  
mkfs(8)  
mklost+found(8)  
mknf(8)  
mknod(8)  
mkpasswd(8)  
mount(8)  
named(8)  
ncheck(8)  
newsfs(8)  
newst(8)  
nfsaccess(8)  
nfarchive(8)  
nfdump(8)  
nfload(8)  
nfmail(8)  
nfrcv(8)  
nfxmit(8)  
nu(8)  
off(8)  
on(8)  
op(8)  
opreq\_daemon(8)  
pac(8)  
ping(8c)  
preen(8)  
pstat(8)  
putst(8)  
quot(8)

quotacheck(8)  
quotaoff(8)  
quotaon(8)  
rc(8)  
rdump(8c)  
reboot(8)  
remshent(8)  
renice(8)  
repquota(8)  
restore(8)  
rexecd(8c)  
rivet(8)  
rlogind(8c)  
rmnf(8)  
rmt(8c)  
route(8c)  
routed(8c)  
rrestore(8c)  
rshd(8c)  
rwhod(8c)  
sa(8)  
seestat(8)  
sendmail(8)  
sethdb(8)  
sharecf(8)  
sharer(8)  
shutdown(8)  
slattach(8c)  
spu(8)  
spucmd(8)  
statd(8c)  
sticky(8)  
sumscripts(8)  
swapon(8)  
sync(8)  
sysex(8)  
sysgen(8)  
syslogd(8)  
syspic(8)  
talkd(8)  
telnetd(8c)  
timed(8)  
timedc(8)  
tpconfig(8)  
tpdaemon(8)  
trpt(8c)  
umount(8)  
update(8)  
uucico(8c)  
uuclean(8c)  
uucpd(8c)  
uulook(8)  
uupoll(8c)  
uusnap(8c)  
uuxqt(8c)  
verify(8)  
versatec(8)

vipw(8)  
xdump(8)

---

## Index

---

# Using ConvexOS man pages

---

## What are man pages?

The ConvexOS man pages, online and hardcopy, are a reference for all of the commands available to ConvexOS users. Each command has its own "man page", which may be one or more pages in length.

The ConvexOS man pages are organized into eight sections; commands for similar functions are grouped together. This book contains one of those sections, section eight, which describes tools used primarily by system managers. (Traditionally, section six of the man pages is allotted to games. Because CONVEX does not sell games, the ConvexOS man pages do not contain section six.)

---

## Where to find other ConvexOS man pages

*ConvexOS Man Pages for Users* contains sections one and seven of the ConvexOS man pages. These sections describe:

- Section 1 man pages— commands of general utility and commands for communication with other systems.
- Section 7 man pages — miscellaneous commands, primarily in the areas of text processing and terminal environments.

*ConvexOS Man Pages for Programmers* contains sections two through five of the ConvexOS man pages. These sections describe:

- Section 2 man pages — system calls and error numbers
- Section 3 man pages — various library functions
- Section 4 man pages — special files, related driver functions, and networking support
- Section 5 man pages — the format of various files

---

## How to use ConvexOS man pages

The ConvexOS man pages are available in two formats: the hardcopy man pages contained in this book and the online man pages you can access by using the ConvexOS man command.

---

### Using hard copy man pages

The ConvexOS man pages contained in this book are ordered alphabetically.

---

## Using online man pages

`man` is a program that formats and displays information from the hard copy man pages. When invoked in the simplest way, without any options and without a topic, it displays the corresponding manual page formatted with `nroff`. Access online man pages by entering

```
% man entry-name
```

where *entryname* is the name of the man page to be displayed. You may print hard copies of online man pages by entering

```
% man -t
```

For more information on `man` options, refer to the `man(1)` man page.

---

## Format of man pages

The contents of each man page is divided into subsections (not all of which are relevant to each entry):

NAME	Lists exact name of command or subroutine and describes its purpose
SYNOPSIS	Summarizes use of command or subroutine being described
DESCRIPTION	Discusses use of command or subroutine
FILES	Names files built into the program
SEE ALSO/REFERENCES	Points to related information
DIAGNOSTICS/ERRORS	Discusses diagnostic messages the system produces
BUGS/NOTES	Explains known bugs or deficiencies and any known solutions

---

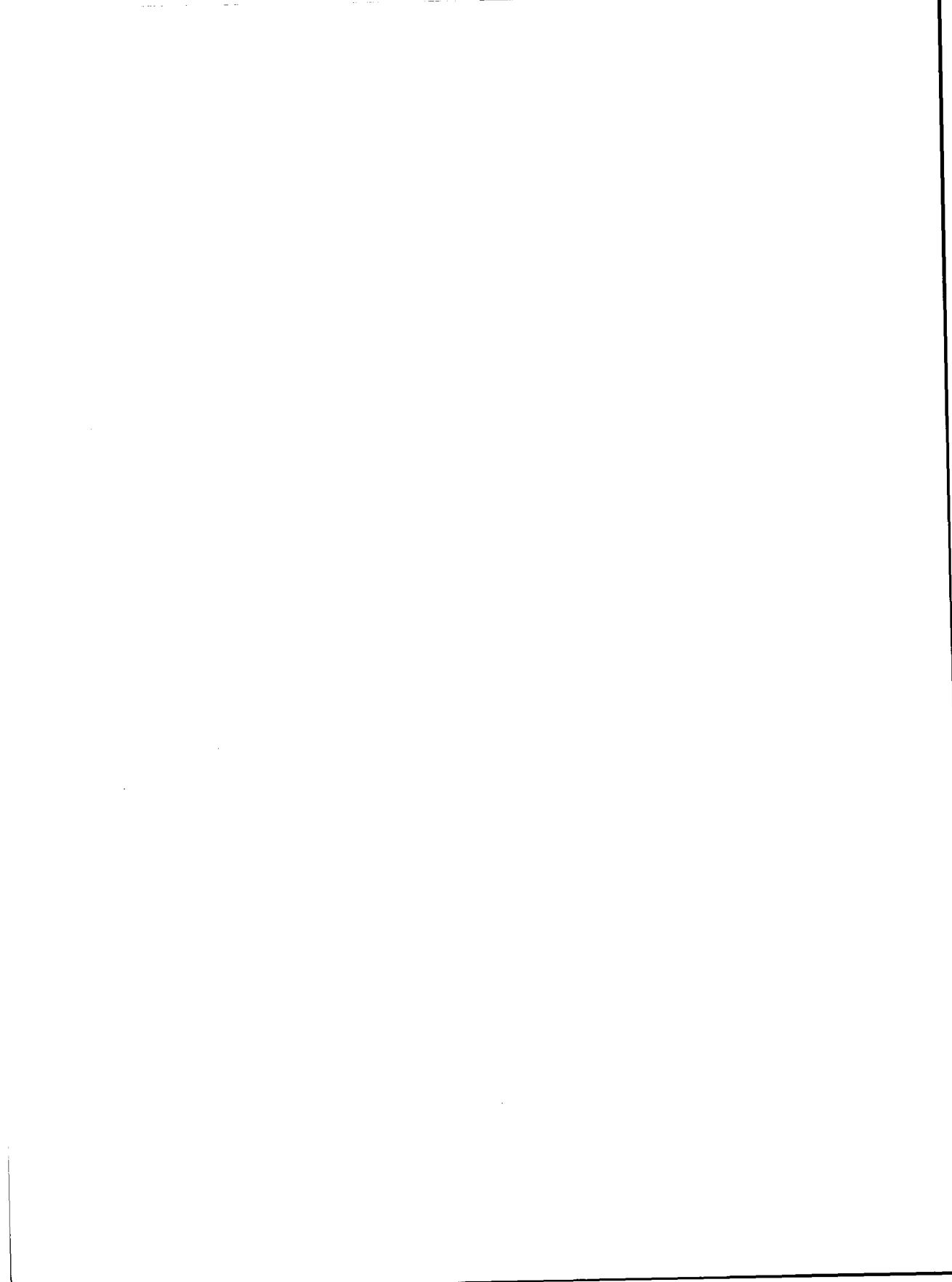
## Technical assistance

If you have questions that are not answered in this book, contact the CONVEX Technical Assistance Center (TAC).

- Within the continental U.S., call 1(800)952-0379.
- From Canada, call 1(800)345-2384.
- All other locations, contact the local CONVEX office.

**Section 8**  
**system management**

---



## NAME

intro – introduction to system maintenance and operation commands

## DESCRIPTION

This section contains information related to system operation and maintenance. In particular, commands used to create new file systems (*newfs*, *newst*, *mkfs*) and to verify the integrity of the file systems (*fack*, *preen*) are described here.

## LIST OF PROGRAMS

<i>Program</i>	<i>Appears on Page</i>	<i>Description</i>
ac	ac(8)	login accounting
accton	sa(8)	system accounting
accounting	accounting(8)	system accounting
actwhocheck	edactwho(8)	edit/check <i>/etc/actwho</i> file
adbccu	adbccu(8)	debugger
arp	arp(8C)	address resolution display and control
atrun	at(1)	execute commands at a later time
biod	nfsd(8)	NFS daemons
catman	catman(8)	create the cat files for the manual
chall	chall(8)	change mode, owner and/or group of a file
chown	chown(8)	change owner
clri	clri(8)	clear i-node
comsat	comsat(8C)	biff server
connecttime	connecttime(8)	compute connect time
cron	cron(8)	clock daemon
ctar	ctar(8)	cartridge tape archiver
dcheck	dcheck(8)	file system directory consistency check
diskuse	diskuse(8)	compute user and group diskspace totals
dump	dump(8)	incremental file system dump
dumpfs	dumpfs(8)	dump file system information
edactwho	edactwho(8)	edit/check <i>/etc/actwho</i> file
edquota	edquota(8)	edit file system quotas
fastboot	fastboot(8)	reboot/halt the system without checking disks
fasthalt	fastboot(8)	reboot/halt the system without checking disks
fsck	fsck(8)	file system consistency check and interactive repair
fsirand	fsirand(8)	randomize inode generation numbers
ftpd	ftpd(8C)	DARPA Internet File Transfer Protocol server
getNAME	makewhatis(8)	rebuild the whatis database
getst	getst(8)	put the stripe table in the kernel
getty	getty(8)	set terminal mode
halt	halt(8)	stop the processor
icheck	icheck(8)	file system storage consistency check
ifconfig	ifconfig(8C)	configure network interface parameters
inetd	inetd(8C)	internet services daemon
init	init(8)	process control initialization
installsw	installsw(8)	create/install software release tapes
lockd	lockd(8C)	network lock manager
lpc	lpc(8)	line printer control program
lpd	lpd(8)	line printer daemon
lpf	lpf(8)	general line printer filter
lprewind	lprewind(8)	line printer rewind
makedbm	makedbm(8)	make a yellow pages dbm file
makedev	makedev(8)	make system special files
makekey	makekey(8)	generate encryption key

makewhatis	makewhatis(8)	rebuild the whatis database
mkfs	mkfs(8)	construct a file system
mklost+found	mklost+found(8)	make a lost+found directory for fsck
mknf	mknf(8)	create and delete notesfiles
mknod	mknod(8)	build special file
mount	mount(8)	mount and dismount file system
mountd	mountd(8C)	NFS mount request server
ncheck	ncheck(8)	generate name from i-numbers
newfs	newfs(8)	construct a new file system
newst	newst(8)	construct a new striped file system
nfacec	nfacec(8)	add access rights to a set of notesfiles
nfarchive	nfarchive(8)	archive notesfiles
nfdump	nfdump(8)	notesfile dump/load program
nload	nfdump(8)	notesfile dump/load program
nmail	nmail(8)	accept mail for a notesfile
nfrcv	nfxmit(8)	notesfile networking programs
nfsd	nfsd(8)	NFS daemons
nfsstat	nfsstat(8)	Network File System statistics
nfxmit	nfxmit(8)	notesfile networking programs
nu	nu(8)	add new user to the system
off	on(8)	enable/disable logins on a tty
pac	pac(8)	printer/plotter accounting information
ping	ping(8C)	send ICMP ECHO_REQUEST packets to a network host
pong	pong(8C)	network debugging
portmap	portmap(8C)	DARPA port to RPC program number mapper
preen	preen(8)	run fsck in parallel over normally mounted disks
pstat	pstat(8)	print system facts
putst	putst(8)	manipulate kernel databases for striped disk partitions
quot	quot(8)	summarize file system ownership
quotacheck	quotacheck(8)	file system quota consistency checker
quotaoff	quotaon(8)	enable/disable file system quotas
quotaon	quotaon(8)	enable/disable file system quotas
rc	rc(8)	command script for auto-reboot and daemons
rdump	rdump(8C)	dump file systems across the network
reboot	reboot(8)	UNIX bootstrapping procedures
renice	renice(8)	alter priority of running processes
repquota	repquota(8)	summarize quotas for a file system
restore	restore(8)	incremental file system restore
rex	rex(8C)	RPC-based remote execution server
rexecd	rexecd(8C)	remote execution server
rlogind	rlogind(8C)	remote login server
rmnf	mknf(8)	create and delete notesfiles
rmt	rmt(8C)	remote magtape protocol server
route	route(8C)	manually manipulate the routing tables
routed	routed(8C)	network routing daemon
rpcinfo	rpcinfo(8)	remote RPC information
rquotad	rquotad(8C)	remote quota server
rrestore	rrestore(8C)	restore a file system dump across the network
rshd	rshd(8C)	remote shell server
rstatd	rstatd(8C)	kernel statistics server
rusersd	rusersd(8C)	network rusers service
rwalld	rwalld(8C)	network rwall server
rwhod	rwhod(8C)	system status server

sa	sa(8)	system accounting
seestat	seestat(8)	print/collect graphs of system status
sendmail	sendmail(8)	send mail over the internet
showmount	showmount(8)	show all remote mounts
shutdown	shutdown(8)	close down the system at a given time
spray	spray(8)	spray packets
sprayd	sprayd(8C)	spray server
spu	spu(8)	read or write SPU UNIX file system
stat	seestat(8)	print/collect graphs of system status
statd	statd(8C)	network status monitor
sticky	sticky(8)	file mode sticky bit
sumscripts	sumscripts(8)	accounting summary awk scripts
swapon	swapon(8)	specify additional device for paging and swapping
sync	sync(8)	update the super block
syssex	syssex(8)	system exerciser that simulates stressful load conditions
sysgen	sysgen(8)	build customized operating system images
syslog	syslog(8)	log systems messages
syspic	syspic(8)	report various system activity in windows
talkd	talkd(8)	talk daemon
telnetd	telnetd(8C)	DARPA TELNET protocol server
tpconfig	tpconfig(8)	configure the tape system
tpd	tpd(8)	tape allocation/deallocation monitoring program
trpt	trpt(8C)	transliterate protocol trace
umount	mount(8)	mount and dismount file system
update	update(8)	periodically update the super block
uucico	uucico(8C)	UUCP protocol handler
uuclean	uuclean(8C)	uucp spool directory clean-up
uulook	uulook(8)	monitor inter-system communications
uupoll	uupoll(8C)	poll a remote system via UUCP
uusnap	uusnap(8C)	show snapshot of the UUCP system
uuxqt	uuxqt(8C)	UUCP command execution
vipw	vipw(8)	edit the password file
ypbind	ypserv(8)	yellow pages server and binder processes
ypinit	ypinit(8)	build and install yellow pages database
ypmake	ypmake(8)	rebuild yellow pages database
yppasswdd	yppasswdd(8C)	server for modifying YP passwd file
yppoll	yppoll(8)	what version of a YP map is at a YP server host
yppush	yppush(8)	force propagation of a changed YP map
ypserv	ypserv(8)	yellow pages server and binder processes
ypset	ypset(8)	point ypbind at a particular server
ypxfr	ypxfr(8)	transfer a YP map from some YP server to here

**NAME**

ac - login accounting

**SYNOPSIS**

`/etc/ac [ -w wtmp ] [ -p ] [ -d ] [ people ] ...`

**DESCRIPTION**

*Ac* produces a printout giving connect time for each user who has logged in during the life of the current *wtmp* file. A total is also produced. `-w` is used to specify an alternate *wtmp* file. `-p` prints individual totals; without this option, only totals are printed. `-d` causes a printout for each midnight to midnight period. Any *people* will limit the printout to only the specified login names. If no *wtmp* file is given, `/usr/adm/wtmp` is used.

The accounting file `/usr/adm/wtmp` is maintained by *init* and *login*. Neither of these programs creates the file, so if it does not exist no connect-time accounting is done. To start accounting, it should be created with length 0. On the other hand if the file is left undisturbed it will grow without bound, so periodically any information desired should be collected and the file truncated.

**FILES**

`/usr/adm/wtmp`

**SEE ALSO**

`init(8)`, `sa(8)`, `login(1)`, `utmp(5)`.

## NAME

accounting – periodic accounting utilities

## SYNOPSIS

```
30 * * * * /usr/adm/accounting > /dev/console
37 00 * * * su root < /usr/lib/diskspace
45 4 * * 5 su root < /usr/lib/diskmail
/usr/adm/accounting -i
/usr/adm/accounting.c
/usr/adm/acctsum.awk
```

## DESCRIPTION

The periodic accounting utilities reside mostly in the */usr/adm* directory and provide simple accounting of disk space and CPU usage for both individuals and complete projects.

*Cron*(1) runs the */usr/adm/accounting* program every hour that the machine is up. If */usr/adm/accounting* finds that the day (or week or month) has changed since the last time it “ran accounting”, then */usr/adm/accounting* invokes the appropriate *daily*, *weekly*, and/or *monthly* script(s). These scripts use *sa*(8) to manipulate a number of incremental and cumulative files.

The command-line syntax that */usr/adm/accounting* uses to invoke the scripts is “*script num old now*”. The *old* and *now* parameters are timestamps as returned by *time*(3). *Old* and *now* indicate when */usr/adm/accounting* last ran the script and what the current time is, respectively. If this is the first time the script has run since accounting was initialized, then *old* is the time of initialization. The value of *num* depends on the script. For the *daily* script, *num* is the current day of the week; Sunday is 0, Monday is 1, etc. For *weekly*, *num* is the current date in the form *month.day.year*; for example, “6.20.83”. For *monthly*, *num* is the current date in the form *month.year*; for example, “6.83”.

The *daily* script copies the */usr/adm/acct* accounting information into a file named */usr/adm/dailyZ* where *Z* is *num* as described above. *Daily* then runs user and process accounting. The accounting accrues in one file on a daily basis, another on a weekly basis, and still another on a monthly basis. The daily user accounting summary is placed into the file */usr/adm/daily.u*. The scripts have the capability of automatically printing this file on the printer (in the middle of the night, usually). The script places the process accounting summary into the */usr/adm/daily.p* file. This file, too, can be printed in the middle of the night if desired.

The weekly and monthly scripts place their output in */usr/adm/week.u*, */usr/adm/week.p*, */usr/adm/month.u*, and */usr/adm/month.p*. These scripts normally print summaries on some printer. The *weekly* script copies cumulative weekly accounting from */usr/adm/usracctw* and */usr/adm/savacctw* into files */usr/adm/u.w.Z* and */usr/adm/s.w.Z* respectively, where *Z* is *num* as described above. Similarly, the *monthly* script copies each month’s accounting into files */usr/adm/u.m.Z* and */usr/adm/s.m.Z*.

The scripts take advantage of the *-F* switch of *sa*(8) which allows the use of alternate accounting files with names like */usr/adm/savacctZ* for accumulation of intermediate totals.

An awk script called */usr/adm/acctsum.awk* defines the various groups into which users’ accounting is summed. Change it by hand unless an automatic regeneration program can deduce groups from some other scheme.

The *-i* switch initializes the times that */usr/adm/accounting* uses in determining when to run daily, weekly, and monthly accounting. These times are kept in file */usr/adm/lastacct*. Use of the *-i* switch will cause daily accounting to run at the beginning of the next day, weekly accounting to run at the beginning of the next week, and monthly accounting to run at the beginning of the next month. This should only be done at installation time or when making a change to the local accounting setup.

A copy of the source for `/usr/adm/accounting` is in `/usr/adm/accounting.c`. The source may be copied and modified to produce a version of `/usr/adm/accounting` which meets local accounting requirements. The *daily*, *weekly*, and *monthly* scripts may also be copied and modified.

The disk accounting runs each day as specified by `/usr/lib/crontab` and produces unsorted (`/usr/adm/disk.today`), sorted (`/usr/adm/disk.today.1`), and grouped (`/usr/adm/disk.today.2`) lists of disk usage. The groupings rely on the `/usr/adm/disksum.awk` script which has the same caveats as the `/usr/adm/acctsum.awk` script mentioned above. The `/usr/lib/diskspace` scripts contain references to the various mounted file systems and hence must be updated when new file systems are created.

Finally, the `/usr/lib/diskmail` script sends users summaries of the recent disk usage as recalled by the `/usr/adm/diskuse/*` files (one for each user). The script relies upon having an up-to-date list of users and their various directories as part of its source.

#### FILES

<code>/usr/adm/daily</code>	daily accounting script
<code>/usr/adm/weekly</code>	weekly accounting script
<code>/usr/adm/monthly</code>	monthly accounting script
<code>/usr/adm/lastacct</code>	timestamps controlling when accounting is run
<code>/usr/adm/accounting</code>	program that executes daily, weekly, monthly scripts
<code>/usr/adm/accounting.c</code>	source for <code>/usr/adm/accounting</code>
<code>/usr/adm/acctsum.awk</code>	awk script for summarizing sa output
<code>/usr/lib/crontab</code>	contains an entry for <code>/usr/adm/accounting</code>

#### SEE ALSO

`cron(1)`, `sa(8)`,  
 "Accounting" chapter in the *CONVEX System Manager's Guide*.

#### BUGS

After accounting is initialized, weekly accounting will run on the first Sunday the machine is up, but if the machine happens to be down all day on this day, accounting will not run until one week after accounting was initialized.

**NAME**

acctconv - accounting file conversion

**SYNOPSIS**

`/usr/adm/acctconv [ source_acctfile ] [ dest_acctfile ]`

**DESCRIPTION**

Convert an existing accounting file to a format suitable for use with ConvexOS V8.0.

With the release of ConvexOS V8.0 the size of one of the fields of the accounting record has increased. This makes V8.0 accounting records incompatible with the accounting records of previous releases. *acctconv* reads records from a ConvexOS V7.1 accounting file, converts the records to V8.0 format, and then writes them to a separate file.

If no *source\_acctfile* or *dest\_acctfile* are specified then */usr/adm/acct* and */usr/adm/acct8.0* are assumed. The size of the *dest\_acctfile* will be larger than the *source\_acctfile*. There must be enough space on the destination file system for the *dest\_acctfile* if the conversion is to work properly.

The ConvexOS installation script saves */usr/adm/acct* as */usr/adm/acct.old* and then executes *acctconv* to convert */usr/adm/acct.old* to */usr/adm/acct*. *acctconv* must be run manually if the accounting data for your system is not in */usr/adm/acct*.

**FILES**

*/usr/adm/acct*                      typical source of accounting data

**SEE ALSO**

acct(5)

“Accounting” chapter in the *CONVEX System Manager's Guide*.

**BUGS**

The *source\_acctfile* and *dest\_acctfile* must have different names.

This utility will not be supported in future releases.

## NAME

adbccu - debugger

## SYNOPSIS

adbccu [-w] [ objfil [ corfil ] ]

## DESCRIPTION

*adbccu* is a general purpose debugging program. It may be used to examine files and to provide a controlled environment for the execution of UNIX programs.

*objfil* is normally an executable program file, preferably containing a symbol table; if not then the symbolic features of *adbccu* cannot be used although the file can still be examined. The default for *objfil* is *a.out*. *corfil* is assumed to be a memory image file produced by *dumpiop* or *dumphsp*, or a character special device */dev/ccu(4)*. See the *CONVEX Student Guide to Writing Device Drivers for the Convex Computer* for additional information on these utilities. If *corfil* is not specified, it isn't opened.

Requests to *adbccu* are read from the standard input and responses are to the standard output. The "-w" options allows writing into *objfil* and *corfil*. *adbccu* ignores QUIT; INTERRUPT causes return to the next *adbccu* command.

In general requests to *adbccu* are of the form

[ *address* ] [ , *count* ] [ *command* ] [ ; ]

If *address* is present then *dot* is set to *address*. Initially *dot* is set to 0. For most commands *count* specifies how many times the command will be executed. The default *count* is 1. *Address* and *count* are expressions.

The interpretation of an address depends on the context it is used in. If a running CCU is being debugged then addresses are interpreted in the usual way in the address space of the CCU. For further details of address mapping see ADDRESSES. When *adbccu* is started on a running CCU, the CCU is suspended. The registers and stack can thus be inspected before any breakpoints are set.

## EXPRESSIONS

- . The value of *dot*.
- + The value of *dot* incremented by the current increment.
- ^ The value of *dot* decremented by the current increment.
- " The last *address* typed.
- integer* A hexadecimal number by default. Decimal numbers begin with "0d". Octal numbers just have a leading zero.
- integer.fraction* A 32 bit floating point number.
- '*cccc*' The ASCII value of up to 4 characters. \ may be used to escape a '.
- < *name* The value of *name*, which is either a variable name or a register name. *adbccu* maintains a number of variables (see VARIABLES) named by single letters or digits. If *name* is a register name then the value of the register is obtained from the system header in *corfil*. The register names are **r0 ... r5 sp pc ps**.
- symbol* A *symbol* is a sequence of upper or lower case letters, underscores or digits, not starting with a digit. The value of the *symbol* is taken from the symbol table in *objfil*. An initial \_ or ~ will be prepended to *symbol* if needed.
- \_ *symbol* In C, the "true name" of an external symbol begins with \_. It may be necessary to utter

this name to distinguish it from internal or hidden variables of a program.

*routine.name*

The address of the variable *name* in the specified C routine. Both *routine* and *name* are *symbols*. If *name* is omitted the value is the address of the most recently activated C stack frame corresponding to *routine*.

(*exp*) The value of the expression *exp*.

**Monadic operators**

\**exp* The contents of the location addressed by *exp* in *corfil*.

@*exp* The contents of the location addressed by *exp* in *objfil*.

-*exp* Integer negation.

~*exp* Bitwise complement.

**Dyadic operators** are left associative and are less binding than monadic operators.

*e1*+*e2* Integer addition.

*e1*-*e2* Integer subtraction.

*e1*\**e2* Integer multiplication.

*e1*%*e2* Integer division.

*e1*&*e2* Bitwise conjunction.

*e1*|*e2* Bitwise disjunction.

*e1*#*e2* *E1* rounded up to the next multiple of *e2*.

**COMMANDS**

Most commands consist of a verb followed by a modifier or list of modifiers. The following verbs are available. (The commands "?" and "/" may be followed by "\*"; see ADDRESSES for further details.)

?*f* Locations starting at *address* in *objfil* are printed according to the format *f*.

/*f* Locations starting at *address* in *corfil* are printed according to the format *f*.

=*f* The value of *address* itself is printed in the styles indicated by the format *f*. (For *i* format "?" is printed for the parts of the instruction that reference subsequent words.)

A *format* consists of one or more characters that specify a style of printing. Each format character may be preceded by a decimal integer that is a repeat count for the format character. While stepping through a format *dot* is incremented temporarily by the amount given for each format letter. If no format is given then the last format is used. The format letters available are as follows.

- o 2 Print 2 bytes in octal. All octal numbers output by *adbccu* are preceded by 0.
- O 4 Print 4 bytes in octal.
- q 2 Print in signed octal.
- Q 4 Print long signed octal.
- d 2 Print in decimal.
- D 4 Print long decimal.
- x 2 Print 2 bytes in hexadecimal.
- X 4 Print 4 bytes in hexadecimal.
- u 2 Print as an unsigned decimal number.
- U 4 Print long unsigned decimal.
- f 4 Print the 32-bit value as a floating point number.
- F 8 Print double floating point.
- b 1 Print the addressed byte in octal.

- c 1** Print the addressed character.
- C 1** Print the addressed character using the following escape convention. Character values 000 to 040 are printed as @ followed by the corresponding character in the range 0100 to 0140. The character @ is printed as @@.
- s n** Print the addressed characters until a zero character is reached.
- S n** Print a string using the @ escape convention. *n* is the length of the string including its zero terminator.
- Y 4** Print 4 bytes in date format (see *ctime(3)*).
- i n** Print as PDP11 instructions. *n* is the number of bytes occupied by the instruction. This style of printing causes variables 1 and 2 to be set to the offset parts of the source and destination respectively.
- a 0** Print the value of *dot* in symbolic form. Symbols are checked to ensure that they have an appropriate type as indicated below.
  - / local or global data symbol
  - ? local or global text symbol
  - = local or global absolute symbol
- p 2** Print the addressed value in symbolic form using the same rules for symbol lookup as **a**.
- t 0** When preceded by an integer tabs to the next appropriate tab stop. For example, **8t** moves to the next 8-space tab stop.
- r 0** Print a space.
- n 0** Print a newline.
- "..." 0** Print the enclosed string.
  - ^ *dot* is decremented by the current increment. Nothing is printed.
  - + *dot* is incremented by 1. Nothing is printed.
  - *dot* is decremented by 1. Nothing is printed.

newline If the previous command temporarily incremented *dot*, make the increment permanent. Repeat the previous command with a *count* of 1.

#### [?/] *value mask*

Words starting at *dot* are masked with *mask* and compared with *value* until a match is found. If **L** is used then the match is for 4 bytes at a time instead of 2. If no match is found then *dot* is unchanged; otherwise *dot* is set to the matched location. If *mask* is omitted then **-1** is used.

#### [?/] **w** *value ...*

Write the 2-byte *value* into the addressed location. If the command is **W**, write 4 bytes. Odd addresses are not allowed when writing to the subprocess address space.

#### [?/] **m** *b1 e1 f1* [?/]

New values for (*b1*, *e1*, *f1*) are recorded. If less than three expressions are given then the remaining map parameters are left unchanged. If the "?" or "/" is followed by "\*" then the second segment (*b2*, *e2*, *f2*) of the mapping is changed. If the list is terminated by "?" or "/" then the file (*objfil* or *corfil* respectively) is used for subsequent requests. (So that, for example, "/m?" will cause "/" to refer to *objfil*.)

> *name dot* is assigned to the variable or register named.

! A shell is called to read the rest of the line following "!".

#### \$*modifier*

Miscellaneous commands. The available *modifiers* are:

- <*f* Read commands from the file *f* and return.
- >*f* Send output to the file *f*, which is created if it does not exist.
- r** Print the general registers and the instruction addressed by **pc**. *dot* is set to **pc**.

- f** Print the floating registers in single or double length. If the floating point status of *ps* is set to double (0200 bit) then double length is used anyway.
- b** Print all breakpoints and their associated counts and commands.
- c** C stack backtrace. If *address* is given then it is taken as the address of the current frame (instead of *r5*). If **C** is used then the names and (16 bit) values of all automatic and static variables are printed for each active function. If *count* is given then only the first *count* frames are printed.
- e** The names and values of external variables are printed.
- w** Set the page width for output to *address* (default 80).
- s** Set the limit for symbol matches to *address* (default 255).
- o** All integers input are regarded as octal.
- d** Reset integer input as described in EXPRESSIONS.
- q** Exit from *adbccu*.
- v** Print all non-zero variables in octal.
- m** Print the address map.

#### *:modifier*

Manage a subprocess. Available modifiers are:

- bc** Set breakpoint at *address*. The breakpoint is executed *count*-1 times before causing a stop. Each time the breakpoint is encountered the command *c* is executed.
- d** Delete breakpoint at *address*.
- c** The CCU is continued. If *address* is given then the CCU is continued at this address. *count* specifies how many breakpoints are to be ignored before stopping.
- s** The same as *c* except that the CCU is single stepped *count* times.

#### VARIABLES

*adbccu* provides a number of variables. Named variables are set initially by *adbccu* but are not used subsequently. Numbered variables are reserved for communication as follows.

- 0** The last value printed.
- 1** The last offset part of an instruction source.
- 2** The previous value of variable 1.

On entry the following are set from the system header in the *corfil*. If *corfil* does not appear to be a **core** file then these values are set from *objfil*.

- b** The base address of the data segment.
- d** The data segment size.
- e** The entry point.
- m** The "magic" number (0405, 0407, 0410 or 0411).
- s** The stack segment size.
- t** The text segment size.

#### ADDRESSES

The address in a file associated with a written address is determined by a mapping associated with that file. Each mapping is represented by two triples (*b1*, *e1*, *f1*) and (*b2*, *e2*, *f2*) and the *file address* corresponding to a written *address* is calculated as follows.

$$b1 \leq \text{address} < e1 \implies \text{file address} = \text{address} + f1 - b1, \text{ otherwise,}$$

$$b2 \leq \text{address} < e2 \implies \text{file address} = \text{address} + f2 - b2,$$

otherwise, the requested *address* is not legal. If a **?** or **/** is followed by an **\*** then only the second triple is used.

The initial setting for *objfil* is suitable for a normal *b.out* file. For crash dumps, the data segment maps the entire file. For running CCUs, the data segment maps the entire 68000 address space so that device registers as well as memory may be accessed.

So that *adbccu* may be used on large files all appropriate values are kept as signed 32-bit integers.

#### FILES

/dev/ccu[3-6]  
a.out  
core

#### SEE ALSO

ccu(4), *CONVEX Student Guide to Writing Device Drivers for the Convex Computer*

#### DIAGNOSTICS

"Adbccu" when there is no current command or format. Comments about inaccessible files, syntax errors, abnormal termination of commands, etc. Exit status is 0, unless last command failed or returned nonzero status.

#### BUGS

It is not possible to set a breakpoint in the MBS communication routines, clock interrupt handler, or *timeout* routines on a running CCU as they are used to deliver the breakpoint message to *adbccu*.

Local variables whose names are the same as an external variable may foul up the accessing of the external.

## NAME

ansidaemon – ANSI-labeled-tape processing filter/daemon

## SYNOPSIS

```
/usr/lib/tape/ansidaemon -l VSN login access physdev
/usr/lib/tape/ansidaemon -u login physdev
/usr/lib/tape/ansidaemon -d RPCprognum physdev reqhost symlink
```

## DESCRIPTION

*ansidaemon* is a program that is only to be invoked by the master daemon of the tape subsystem (*tpdaemon*(8)). This program is almost exclusively an RPC server which will perform I/O requests for ANSI-labeled tapes. It handles all processing of the ANSI magnetic tape labels, the blocking and unblocking of logical records, as well as the translation of *ioctl* commands into a similar functionality for ANSI tapes (where possible). With ANSI-labeled tapes, attempts to read beyond the last file in the fileset yields the error ENOENT “No such file or directory”.

CONVEX conforms roughly to Level 3 of the *ANSI X3.27-1978* specification. However, there is currently no way to specify or examine user labels (UVL, UHL, and UTL). In addition to the fixed (F) and variable (D) length record formats that are defined within the ANSI specification, there is also support for the old U (“undefined”) format. CONVEX currently defines this to operate as raw tape would; that is, for each *read* or *write* system call, one physical block is read from or written to tape, without any interpretation of the data included in that block. This format seems useful in creating a binary file that is surrounded by ANSI labels.

The label (-l) and unlabel (-u) options are designed to operate only on one physical tape volume, but the driver (-d) mode processes I/O requests for an entire ANSI tapeset (possibly more than one physical tape).

Options:

- l Labels an unlabeled tape. The first three arguments are used in creating the initial VOL1 label for the tape.
  - VSN* is a string (up to six characters) to be used as the unique volume identifier, or volume serial number.
  - login* is the login name of the user who invoked *tplabel*(1). It is converted into uppercase and used as the owner of the newly initialized ANSI volume.
  - access* is a single character that is placed directly into the VOL1 accessibility field. It should be a blank if the volume is not to have restricted access; this is the default condition.
  - physdev* specifies the physical path of the drive where the tape has been mounted.
- u Unlabels an ANSI-labeled tape. Only the owner of a tape or the superuser is allowed to execute the *tpunlabel*(1) command, which results in the *ansidaemon* being invoked with this option.
  - login* is the login name of the user running *tpunlabel*(1). It is converted to uppercase before being compared with the owner that has been stored in the tape’s VOL1 label when verify authorization to unlabel the mounted tape.
  - physdev* specifies the physical tape drive that contains the tape being unlabeled.
- d Invokes the daemon in driver mode. The daemon is automatically started by *tpdaemon*(8) once a labeled tape has been mounted. There is one *ansidaemon* running per active (mounted) ANSI tapeset. The *ansidaemon* program then serves as a pseudo device driver which will accept *open*, *close*, *read*, *write*, and *ioctl* calls from the kernel and translate each request into its ANSI equivalent. Labels are transparently written on

output, and verified on input. Existing programs such as *cp(1)* and *cat(1)* are able to write ANSI-labeled tapes “transparently” using this interface. The required arguments are listed below:

*RPCprogram*

is the RPC(3) program number of this invocation of the *ansidaemon*.

*physdev*

as before, this is the path to the physical tape device where the first tape in the ANSI tapeset is mounted.

*reqhost* is the hostname of the machine originating the request to use the device.

*symlink* is the symbolic link name associated with this ANSI tapeset (given by the user in the *tpmount(1)* command).

SEE ALSO

*tpmount(1)*, *tpunmount(1)*, *tpattr(1)*, *tplabel(1)*, *tpunlabel(1)*, *tpqueue(1)*, *tpwait(1)*, *tape(3)*, *tpconfig(8)*, *tpdaemon(8)*, *mt(1)*

*ANSI X3.27-1978*: Magnetic Tape Labels and File Structure for Information Interchange.

## NAME

arp – address resolution display and control

## SYNOPSIS

```
arp hostname
arp -a [ vmuniz ] [ kmem ]
arp -d hostname
arp -s hostname ether_addr [ temp ] [ pub ] [ trail ]
arp -f filename
arp -e filename
```

## DESCRIPTION

The *arp* program displays and modifies Internet-to-Ethernet address translation tables used by the address resolution protocol (*arp*(4p)).

With no flags, the program displays the current ARP entry for *hostname*. The host may be specified by name or by number, using Internet dot notation. With the **-a** flag, the program displays all of the current ARP entries by reading the table from the file *kmem* (default */dev/kmem*) based on the kernel file *vmuniz* (default */vmuniz*).

With the **-d** flag, a super-user may delete an entry for the host called *hostname*.

The **-s** flag is given to create an ARP entry for the host called *hostname* with the Ethernet address *ether\_addr*. The Ethernet address is given as six hex bytes separated by colons. The entry will be permanent unless the word **temp** is given in the command. If the word **pub** is given, the entry will be “published”; i.e., this system will act as an ARP server, responding to requests for *hostname* even though the host address is not its own. The word **trail** indicates that trailer encapsulations may be sent to this host.

The **-f** flag causes the file *filename* to be read and multiple entries to be set in the ARP tables. Entries in the file should be of the form

```
hostname ether_addr [ temp ] [ pub ]
```

with argument meanings as given above.

The **-e** flag causes the file *filename* to be read and multiple entries to be set in the ARP tables. Entries in the file should be of the form

```
ether_addr hostname [ temp ] [ pub ] [ trail ]
```

with argument meanings as given above. Note that this flag is the same as the **-f** flag except that the first two fields in the file to be read are reversed. This format is suitable for reading the */etc/ethers* file used by the yellow pages.

## FILES

<i>/vmuniz</i>	Kernel name list
<i>/dev/mem</i>	Kernel data values
<i>/etc/ethers</i>	For permanent arp entries
<i>/lib/kernsyms/symdata*</i>	Kernel symbol addresses

## SEE ALSO

inet(3N), arp(4P), ethers(5), ifconfig(8C)

## NOTES

*arp* is an optional product; for more information, contact your CONVEX sales representative.

**NAME**

*avail* - maintain and process system availability information

**SYNOPSIS**

```
avail [ -m addr ] [ -t ]
```

**DESCRIPTION**

*avail* maintains, processes, and delivers availability information on CONVEX systems. */usr/spool/convex/avail* should only be invoked automatically from root's crontab at 15 minute intervals. To activate the availability reporting system, the following line should be added to */usr/lib/crontab*

```
00,15,30,45 * * * * /usr/spool/convex/avail
```

and the following line should be added to */etc/rc.local*

```
/usr/spool/convex/reboot_script < /dev/console > /dev/console
```

When activated, the system will write a status line to */usr/spool/convex/availlog* containing local time, number of users, and 15 minute load average. At multi-user reboot, information is logged to */usr/spool/convex/reboot\_log*. This data includes the local time, the results of a call to *sysinfo(2)*, and information provided to *shutdown(8)*, if any.

If the file, */usr/spool/convex/avail.conf* exists, it's contents, in addition to an option to input free-form text, are used to generate a menu of shutdown reasons at reboot. This menu is displayed and *avail(8)* will wait for a response for 2 minutes before timing out and continuing the reboot. If a reason is chosen, it is added to */usr/spool/convex/reboot\_log*. This file contains a set of default reasons, but may be edited by system administrators.

At 00:00:00 each Sunday morning, the following SPU files are compared with those from the previous week and differences are recorded:

```
/mnt/usr/scn/cop.out
/mnt/DIAG_DB_REV
/mnt/DIAG_REV
/ioconfig
/mnt/usr/scn/cop.mem
/mnt/usr/lib/softlog
/UNIX_REV
/mnt/errlog
/mnt/usr/ucode/UCODE_REV
```

These differences, */usr/spool/convex/reboot\_log*, and */usr/spool/convex/availlog* are either mailed to an address at CONVEX, or archived in a *tar(1)* format, according to options below.

The command line options to *avail(8)* may be added to the line in */usr/lib/crontab*. The **-m** option specifies an alternate mail address for availability data. By default, the data is mailed to "convex!avail".

The **-t** option specifies that availability data be archived using *tar(1)*. If this option is used, a file with the following format will be produced.

```
/usr/spool/convex/avail.m.d.y
```

Where m = month, d = day, and y = year.

**SEE ALSO**

*shutdown(8)*, *sysinfo(2)*, *tar(1)*

**NAME**

*catinfo* - create the formatted files for the info system

**SYNOPSIS**

**/usr/convex/catinfo**

**DESCRIPTION**

*catinfo* creates the preformatted versions of the on-line info system screens from the *nroff* input files. Each command description file used by the CONVEX Information System is successively run through *nroff*, and the resulting formatted output is placed in a corresponding file in a separate subdirectory.

The input files, each named according to the command it describes, are found in */usr/infosys/screens*. The output files, whose names are formed by appending a '.f' to the input filename, are placed in */usr/infosys/fscreens*.

Similarly, the information screens provided with any optional products your site might have, are also processed by *nroff*. These optional input files are found in */usr/infosys/optscreens*, with the corresponding formatted output files being placed in */usr/infosys/optfscreens*.

**FILES**

<i>/usr/infosys/screens/*</i>	raw ( <i>nroff</i> input) information screens
<i>/usr/infosys/fscreens/*.f</i>	formatted ( <i>nroff</i> output) information screens
<i>/usr/infosys/optscreens/*</i>	raw ( <i>nroff</i> input) optional information screens
<i>/usr/infosys/optfscreens/*.f</i>	formatted ( <i>nroff</i> output) optional information screens

**SEE ALSO**

*info(1)*

**NAME**

`catman` - create the cat files for the manual

**SYNOPSIS**

`/etc/catman [ -p ] [ -n ] [ -w ] [ sections ]`

**DESCRIPTION**

*Catman* creates the preformatted versions of the on-line manual from the nroff input files. Each manual page is examined and those whose preformatted versions are missing or out of date are recreated. If any changes are made, *catman* will recreate the `/usr/lib/whatis` database.

If there is one parameter not starting with a '-', it is interpreted to be a list of manual sections in which to look. Each character in that list can be the name of any "chapter" in the `/usr/man` directory, where each character *X* specifies the subdirectory `manX`. The specified sections do not have to be numeric. For example

**catman 1231**

will cause the updating to only happen to manual sections 1, 2, and 3, and the `/usr/man/man1` "chapter."

Options:

- n prevents creations of `/usr/lib/whatis`.
- p prints what would be done instead of doing it.
- w causes only the `/usr/lib/whatis` database to be created. No manual reformatting is done.

**FILES**

<code>/usr/man/man?/*.*</code>	raw (nroff input) manual sections
<code>/usr/man/cat?/*.*</code>	preformatted manual pages
<code>/usr/lib/makewhatis</code>	commands to make whatis database

**SEE ALSO**

`man(1)`

**NAME**

chall - change mode, owner and/or group of a file

**SYNOPSIS**

**chall** [ -m *mode* ] [ -g *group* ] [ -o *owner* ] [ -l ] *filename...*

**DESCRIPTION**

*Chall* changes the access mode, owner-ID, and group-ID of the list of *filenames* to the values specified for *mode*, *owner* and *group*. Any combination of -l, -m, -o, and -g may be specified. If a *filename* is a directory then the modifications are made recursively in that directory. The *group* may be either a decimal group ID or a name found in the */etc/group* file. The *owner* may be either a decimal user ID or a name found in the */etc/passwd* file. The access *mode* may be either an octal value or a mode string as described in *chmod(1)*.

The user invoking *chall* must be the super-user.

The -l flag specifies that action on symbolic links, both to normal files and directories, be allowed. Without this flag, links encountered will have -o and -g changes applied, but not -m. With this flag, -m changes, along with directory traversal, are allowed.

**FILES**

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

**SEE ALSO**

chgrp(1), chmod(1), group(5), passwd(5), chown(8)

## NAME

charge – change charges for share system usage calculations

## SYNOPSIS

**charge** [*flags*] [*percent*]

## DESCRIPTION

*charge* will change the costs associated with the usage calculations for the Share scheduler. The **-C** flag, if used, should be mentioned first to cause *charge* to charge the charging values currently in place instead of the default values. *charge* with no flags will install the default values that are defined in `<sys/charges.h>` The optional flags are as follows:

- C** Causes *charge* to read in the current values, which it will use instead of the defaults. **Must be first flag if used.**
- D*h1,h2*** Will set the decay rate for process priorities with normal *nice* so that they will decay to half their initial value in *h1* seconds, and set the decay rate for process priorities with maximum *nice* so that they will decay to half their initial value in *h2* seconds. *h1* is required to be less than *h2*.
- E*half-life*** Will set the decay rate for users' process *run rates* so that they will decay to half their initial value in *half-life* seconds. *half-life* must be greater than 0.
- F*flags*** Sets various global scheduling flags — see *share(5)* for details. *flags* are assumed to be in octal.
- G*maxgroups*** Sets the maximum depth for the scheduling tree. *maxgroups* must be greater than or equal to zero.
- K*half-life[s]*** Will set the decay rate for users' usages so that they will decay to half their initial value in *half-life* hours. If the *half-life* is followed by the character *s*, then the number will be interpreted as seconds. *half-life* must be a positive integer less than 25842.
- P*maxpri*** Absolute upper bound for a process's priority. (Something less than 1/128th of the largest non-negative integer.)
- Q*maxupri*** Upper bound for normal processes' priorities. *Idle* processes run with priorities in the range *maxpri* < *pri* < *maxupri*.
- R*delay*** Sets the update delay in seconds for the *rates* program when the *rates* program is in continuous display mode (**-c** option).
- S*maxusers*** Sets the maximum number of users and groups that can be active. Note that this cannot exceed the maximum configured in the kernel. (The maximum number configured in the kernel will be used as the default value if it is smaller than the value in `<sys/charges.h>`) *maxusers* must be greater than or equal to zero.
- U*maxusage*** Upper bound for "reasonable" usages. Users with usages larger than this are grouped together and given a *normalized usage* which prevents them from interfering with "normal" users. *maxusage* must be greater than zero.
- X*maxushare*** If the LIMSHARE scheduling flag is on, then this parameter limits the maximum effective share an individual user can have to *maxushare* times their allocated share.
- Y*mingshare*** If the ADJGROUPS scheduling flag is on, and any group is getting less than *mingshare* times its allocated share, ADJGROUPS adjusts the run rates of groups getting more than their share up, decreasing their priority.
- Z*mazrate*** If the ADJGROUPS scheduling flag is on, then this number is the maximum amount by which a group member may be adjusted.

<b>-bbio</b>	The charge for a disk block I/O operation. <i>bio</i> must be greater than or equal to zero.
<b>-mclick</b>	The charge for a memory tick. <i>click</i> must be greater than or equal to zero.
<b>-ssyscall</b>	The charge for a system call. <i>syscall</i> must be greater than or equal to zero.
<b>-ttick</b>	The charge for a CPU tick. <i>tick</i> must be greater than or equal to zero.
<b>-ytio</b>	The charge for a stream I/O operation. (This is really dependent on the number of kernel buffer operations, so a <i>write(1)</i> will cost the same as a <i>write(64)</i> to an ordinary stream, or a <i>write(1024)</i> to a pipe.) <i>tio</i> must be greater than or equal to zero.
<b>percent</b>	The percentage change to apply to all the charges. If any of the optional flags are used but no value is specified, the parameter will be set to 0, the default value.

**EXAMPLES**

To change the costs to 10% of the current values:

**charge -C 10**

**FILES**

/usr/include/sys/charges.h	The default values in the kernel.
/usr/include/sys/share.h	Definition of charges structure.

**SEE ALSO**

charges(1), pl(1), pwrite(1), rates(1), inode(5), share(5), shares(5), lim(8), remshent(8), sharer(8)

**BUGS**

The percent flag will also affect any new constants, so bias them accordingly.

The defaults are hardly ever relevant.

**NOTES**

*charge* is part of the Share scheduler. Share is an optional product; contact your CONVEX sales representative for more information.

**NAME**

chown - change owner

**SYNOPSIS**

*/etc/chown* [ **-f** ] owner file ...

**DESCRIPTION**

*Chown* changes the owner of the *files* to *owner*. The owner may be either a decimal UID or a login name found in the password file.

Only the super-user can change owner, in order to simplify accounting procedures. No errors are reported when the **-f** (force) option is given.

**FILES**

*/etc/passwd*

**SEE ALSO**

chgrp(1), chown(2), passwd(5), group(5)

**NOTES**

If *chown* is performed on a symbolic link, the modifications will take place on the symbolic link and not on the file pointed to by the link.

**NAME**

*clri* - clear i-node

**SYNOPSIS**

*/etc/clri* filesystem i-number ...

**DESCRIPTION**

**N.B.:** *Clri* is obsoleted for normal file system repair work by *fsck(8)*.

*Clri* writes zeros on the i-nodes with the decimal *i-numbers* on the *filesystem*. After *clri*, any blocks in the affected file will show up as 'missing' in an *fsck(8)* of the *filesystem*.

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.

**SEE ALSO**

*fsck(8)*

**BUGS**

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

**NAME**

comsat, in.comsat - biff server

**SYNOPSIS**

`/usr/etc/in.comsat [ -t timeout ]`

**DESCRIPTION**

*comsat* is the server process which listens for reports of incoming mail and notifies users if they have requested this service. It is invoked as needed by *inetd*(8C), and times out if inactive for two minutes. The `-t` option can be specified to use a different timeout length than the default. The timeout length is specified in minutes.

*comsat* listens on a datagram port associated with the "biff" service specification (see *services*(5)) for one line messages of the form

`user@ mailbox-offset`

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

"localhost" must be defined in */etc/hosts* in order for *comsat* to work properly.

**FILES**

*/etc/utmp* contains data on who is logged in to what terminals.

**SEE ALSO**

*biff*(1), *inetd*(8C)

**BUGS**

The message header filtering is prone to error.

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

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

## NAME

connecttime - compute connect time from wtmp and bill-acct

## SYNOPSIS

**connecttime** [ **-o** *time* ] [ **-n** *time* ] [ **-w** *wtmpfile* ] [ **-b** *billfile* ]

## DESCRIPTION

*connecttime* takes the log files generated by *login* and *bill* and merges them. The output of *connecttime* is typically used as the input to the *connecttime.awk* script, which is described in *sumscripts(8)*.

*connecttime* produces two types of entries. The first type begins with the keyword "started" and indicates either a login or a user billing to a new account. The second type begins with the keyword "terminated" and indicates either a logout or a user ending billing to an account. Note that the execution of a single *bill* command will show up as two entries: one for the termination of billing to the old account and one for the initiation of billing to the new account.

The formats of these two types are described below.

```

started      time  tty   uid   gid   aid   remote
terminated   time  tty

```

"Started" and "terminated" appear as text. *time* is the time as returned by *time()*; *tty*, *uid*, *gid*, and *aid* are self-explanatory. If the user logged on from another system, this will show up in the *remote* field.

The command-line options are:

- b** *billfile*      Specify a bill log file to use instead of the default file */usr/adm/bill-acct*.
- n** *time*          Specify the time of the newest entry. Entries made in the log files after this time will be ignored.
- o** *time*          Specify the time of the oldest entry. Entries made in the log files before this time will be ignored.
- w** *wtmpfile*      Specify a login log file to use instead of the default file */usr/adm/wtmp*.

## NOTE

The *awk* script *connecttime.awk* counts remote logging in its *connecttime* calculation. If this is not desired, a locally modified version of *connecttime.awk* may be used, that will compute connect time in a way that is acceptable to a given installation.

## SEE ALSO

*bill(1)*, *time(3c)*, *bill-acct(5)*, *utmp(5)*, *sumscripts(8)*,  
 "Accounting" chapter in the *CONVEX System Manager's Guide*.

## BUGS

*connecttime* ignores date change entries in *wtmpfile*.

**NAME**

cpuconf - CPU Configuration

**SYNOPSIS**

```
/etc/cpuconf [-e [cpuid | all] -d [cpuid | all] ]
```

**DESCRIPTION**

If no arguments are given, the current cpu configuration is printed. If a cpuid is specified, the specified cpu is either enabled or disabled. The *-e* flag is used to enable UNIX process scheduling on a specified cpu. The *-d* flag is used to disable UNIX process scheduling on a specified cpu. For example, given a 4 cpu system:

```
cpuconf -d0 -e 2 -d1
```

Disables process scheduling on cpus 0 and 1, and enables process scheduling on cpu 2. Process scheduling on cpu 3 remains unchanged.

The command:

```
cpuconf -d all
```

will disable all but one cpu in the system.

This command can be executed at any time without any adverse effects on the systems operation. The disabling of a cpu only inhibits UNIX scheduling of the CPU and does not disable interrupt processing.

**DIAGNOSTICS**

'cpuconf: cpuconf set failed Not owner' if you try to change the cpu configuration but are not the superuser.

'cpuconf: Attempt to turn off all CPUs' if you try to disable every cpu in the system.

## NAME

crashdump – write a crash dump to tape

## SYNOPSIS

```
crashdump [-DsuH] [-C cop.out] [-I ioconfig] [-v vmunix] [-i iop] [-h hsp] [-V viop] [-a
acm201.x00] [-d idc] [-L bootcmd.local] [-B bootcmd] [-T tunables]
```

## DESCRIPTION

Taking a crash dump is recommended after a ConvexOS crash or hang, as it may provide the data necessary to determine and fix the cause of the problem.

*crashdump* writes a crash dump to a local tape drive. *crashdump* is executed from the SPU of a machine whose JP is down and should not be executed while the JP is running.

A predetermined set of files is transmitted whose default path names are */mnt/usr/scn/cop.out*, */ioconfig*, *vmunix*, *iop*, *hsp*, *viop*, *acm201.x00*, *idc*, *bootcmd.local*, *bootcmd*, and *tunables*. These files may be specified explicitly by using the *-C*, *-I*, *-v*, *-i*, *-h*, *-V*, *-a*, *-d*, *-L*, *-B*, and *-T* switches, respectively. If these files cannot be found in the current directory, *crashdump* changes its current directory to */mnt/os* and tries again; if this fails *crashdump* itself fails with an error message. *crashdump* will also fail if any of the temporary files that it creates in the current directory already exist, as will be the case if it exits abnormally without cleaning up.

The *-D* switch enables the output of debug information. (Note: in previous releases, this was the *'-d'* switch, which is now used for the optional specification of the IDC disk controller executable pathname.)

The *-s* switch forces the crash dump to be written on the SPU cartridge tape drive.

The *-u* switch disables unloading of the tape after the crash dump is complete.

The *-H* switch will invoke the hardware dump facility on the SPU. It generates an output file, *hwdump.out*, which will be included with the other miscellaneous files in the *tar* format section of the tape.

*crashdump* determines whether or not the machine contains a tape drive by looking in */ioconfig*. The default behavior is to dump to a JP tape drive if such a tape drive exists. If not, the *-s* switch must be used to write to the SPU cartridge tape drive. If */ioconfig* contains more than 1 tape controller, *crashdump* will use the entry which appears first in */ioconfig*.

## SEE ALSO

crashdump(5), crashread(8), inetd(8c), arp(8c), route(8c), routed(8c)

## NAME

crashread – read physical memory from a crash dump tape

## SYNOPSIS

```
crashread [-f tape_device] [-H] [-m mem_start-mem_end] .. [-i iop_num] .. [-v viop_num] .. [-h
hsp_num] .. [-d idc_num] .. [-M]
```

## DESCRIPTION

*crashread* is used to read data from a crash dump tape. Files are created in the current directory to hold the dump data. When reading a dump tape, it is best to create a new directory and make the new directory the current directory before invoking *crashread*.

Crash dump tapes are stored on 6250 bpi tapes. The 6250 bpi tape device that the tape is mounted on is specified with the *-f* switch. The default is */dev/rmt20*.

The default action of *crashread* is to read everything on the tape(s). Specific items may be requested by command-line switches. If any specific items are requested via arguments, then only those items specifically requested are read, except the tape's header file, which *crashread* reads every time it accesses a tape. For crash dumps occupying more than one tape, the user will be prompted to mount the appropriate tape before anything is read.

Items may be specifically requested using the following switches:

- h hsp\_num*      Memory image of HSP number *hsp\_num*. *hsp\_num* must be between 3 and 7 on C1 or 0 and 7 on C2.
- H*              Header file.
- i iop\_num*      Memory image of IOP number *iop\_num*. *iop\_num* must be between 3 and 7 on C1 or 0 and 7 on C2.
- v viop\_num*     Memory image of VIOP number *viop\_num*. *viop\_num* must be between 0 and 3 on C1 or 0 and 7 on C2.
- i idc\_num*      Memory image of IDC number *idc\_num*. *idc\_num* must be between 0 and 7 on C2. (IDC is not available on C1).
- m mem\_start-mem\_end*  
Memory locations *mem\_start* to *mem\_end*, specified in hex. *mem\_start* is rounded down to the nearest multiple of 2 MB; *mem\_end* is rounded up to the nearest multiple of 2 MB, then 1 is subtracted from it. For example, *-m 1-2* specifies that memory locations hex 0 to hex 1ffff be read. *crashread* compresses zero-filled pages and holes in the PCM using *lseek(2)* to save disk space.
- M*              The tar file at the end of the first tape containing miscellaneous files such as *ioconfig* and *vmunix*.

*crashread* creates a file called *dumpstatus* in the current directory. This file contains a list of some items that are on the crash dump tape, and whether or not these items have been read. *crashread* checks this *dumpstatus* and only reads data that has not yet been read according to *dumpstatus*.

The 64K MBS memory file mentioned in *crashdump(5)* is treated by *crashread* the same as the first 64K of main memory. Whenever *crashread* reads the first 2MB of memory, it also reads the MBS memory file and writes it into the location in *mm.core* corresponding to the real MBS memory area. (This location happens to be location 0.)

## FILES

*Crashread* creates the following files in the current directory:

- bootcmd*        *bootcmd* from the SPU
- bootcmd.local* *bootcmd.local* from the SPU, if found
- commregs*      Communication registers for a C2

cop.out	<i>/mnt/usr/scn/cop.out</i> from the SPU
dumpstatus	Status of read from tape
errlog	<i>/mnt/errlog</i> from the SPU.
headerX	Header for tape number X.
hsp	HSP executable.
hsp.coreX	Core image for HSP number X.
hwdump.out	Output from hardware dump, if taken
ioconfig	<i>ioconfig</i> from the SPU.
iop	IOP executable.
iop.coreX	Core image for IOP number X.
viop	VIOP executable
viop.coreX	Core image for VIOP number X.
idc	IDC executable
idc.coreX	Core image for IDC number X.
mm.core	Main memory core image.
registers	CPU registers
	The <i>registers</i> file will be present <b>only</b> if a hardware dump was not taken. If a hardware dump was taken, <i>hwdump.out</i> will contain the CPU registers.
tunables	<i>tunables</i> from the SPU
vmunix	<i>vmunix</i> from the SPU.

**WARNING**

It is easy to make the mistake of reading different crash dumps into the same directory. Check the contents of the current directory before using *crashread*.

**SEE ALSO**

dd(1), mt(1), tar(1), tpmount(1), tpmount(1), crashdump(5), crashdump(8)

## NAME

*ctar* – cartridge tape archiver

## SYNOPSIS

**ctar** [*key*] [*name ...*]

## DESCRIPTION

*ctar* saves and restores multiple files on a single file SPU cartridge tape. It is functionally identical to *tar* except that the only permitted archive device is the SPU cartridge tape. *ctar*'s actions are controlled by the *key* argument. The *key* is a string of characters containing at most one function letter and possibly one or more function modifiers. Other arguments to *ctar* are file or directory *names* specifying which files to dump or restore. In all cases, appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

To use *ctar*, one's user ID must be that of the superuser.

The function portion of the *key* is specified by one of the following letters:

- r** The named files are written on the end of the tape. The **c** function implies this.
- x** The named files are extracted from the tape. If the named file matches a directory whose contents had been written onto the tape, this directory is (recursively) extracted. The owner, modification time, and mode are restored (if possible). If no file argument is given, the entire content of the tape is extracted. Note that if multiple entries specifying the same file are on the tape, the last one overwrites all earlier.
- t** The names of the specified files are listed each time they occur on the tape. If no file argument is given, all of the names on the tape are listed.
- u** The named files are added to the tape if either they are not already there or have been modified since last put on the tape.
- c** Create a new tape; writing begins on the beginning of the tape instead of after the last file. This command implies **r**.
- o** On output, *ctar* normally places information specifying owner and modes of directories in the archive. Former versions of *ctar*, when encountering this information will give error message of the form  
     “<name>/: cannot create”  
 This option will suppress the directory information.
- p** This option says to restore files to their original modes, ignoring the present *umask*(2). Setuid and *sticky*(8) information will also be restored to the superuser.

The following characters may be used in addition to the letter which selects the function desired.

- v** Normally *ctar* does its work silently. The **v** (verbose) option make *ctar* type the name of each file it treats preceded by the function letter. With the **t** function, the verbose option gives more information about the tape entries than just their names.
- w** *ctar* prints the action to be taken followed by filename, then wait for user confirmation. If a word beginning with 'y' is given, the action is done. Any other input means don't do it.
- f** *ctar* uses the next argument as the name of the archive instead of */dev/rct0e*. Note that the default device on a C2 is */dev/rmt0* or */dev/rmt1*. If the name of the file is -, *ctar* writes to standard output or reads from standard input, whichever is appropriate. Thus, *ctar* can be used as the head or tail of a filter chain. *ctar* can also be used to copy hierarchies with the command:

```
cd fromdir; ctar cf - . | (cd todir; ctar xf -)
```

- b** *ctar* uses the next argument as the blocking factor for tape records. The default is 238. The block size is determined automatically when reading tapes (key letters **x** and **t**)
- l** Tells *ctar* to complain if it cannot resolve all of the links to the files dumped. If this is not specified, no error messages are printed.
- m** Tells *ctar* not to restore the modification times. The modification time will be the time of extraction.
- h** Forces *ctar* to follow symbolic links as if they were normal files or directories. Normally, *ctar* does not follow symbolic links.
- B** Forces input and output blocking to 238 blocks per record. This option was added so that *ctar* can work across a communications channel where the blocking may not be maintained.
- i** Causes *ctar* to ignore directory checksum errors while reading.

If a filename is preceded by **-C**, then *ctar* will perform a *chdir(2)* to that filename. This allows multiple directories not related by a close common parent to be archived using short relative path names. For example, to archive files from */usr/include* and from */etc*, one might use

```
ctar c -C /usr include -C / etc
```

Previous restrictions dealing with *ctar*'s inability to properly handle blocked archives have been lifted.

#### FILES

```
/dev/rct0? (SPU device)
/dev/rmt?
/tmp/tar*
```

#### SEE ALSO

```
tar(1), su(1)
```

#### DIAGNOSTICS

Complaints about bad key characters and tape read/write errors.

Complaints if enough memory is not available to hold the link tables.

#### BUGS

There is no way to ask for the *n*th occurrence of a file.

Tape errors are handled ungracefully.

The **u** option can be slow.

The current limit on filename length is 100 characters.

There is no way to selectively follow symbolic links.

**NAME**

`dcheck` - file system directory consistency check

**SYNOPSIS**

`/etc/dcheck [ -i numbers ] [ filesystem ]`

**DESCRIPTION**

**N.B.:** *Dcheck* is obsoleted for normal consistency checking by *fsck*(8).

*Dcheck* reads the directories in a file system and compares the link-count in each i-node with the number of directory entries by which it is referenced. If the file system is not specified, a set of default file systems is checked.

The `-i` flag is followed by a list of i-numbers; when one of those i-numbers turns up in a directory, the number, the i-number of the directory, and the name of the entry are reported.

The program is fastest if the raw version of the special file is used, since the i-list is read in large chunks.

**FILES**

Default file systems vary with installation.

**SEE ALSO**

*fsck*(8), *icheck*(8), *fs*(5), *clri*(8), *ncheck*(8)

**DIAGNOSTICS**

When a file turns up for which the link-count and the number of directory entries disagree, the relevant facts are reported. Allocated files which have 0 link-count and no entries are also listed. The only dangerous situation occurs when there are more entries than links; if entries are removed, so the link-count drops to 0, the remaining entries point to thin air. They should be removed. When there are more links than entries, or there is an allocated file with neither links nor entries, some disk space may be lost but the situation will not degenerate.

**BUGS**

Since *dcheck* is inherently two-pass in nature, extraneous diagnostics may be produced if applied to active file systems.

*Dcheck* is obsoleted by *fsck* and remains for historical reasons.

**NAME**

diskuse – compute user and group disk-space totals

**SYNOPSIS**

diskuse [-i] [-d *dir*]

**DESCRIPTION**

*Diskuse* recursively visits every file within a given directory and computes kilobyte totals for each user ID and group ID that owns files in the hierarchy. The top of the directory hierarchy is assumed to be the current directory, unless an alternate directory is specified with the **-d** option. If the **-i** option is given, *diskuse* ignores any NFS mounted directories in the hierarchy. Using the **-i** option will **significantly** increase the time it takes to run *diskuse*.

The output of *diskuse* is in a form suitable for processing by the *disk\** awk scripts described in *sumscripts(8)*.

**SEE ALSO**

du(1), sumscripts(8),  
“Accounting Reports” chapter in *Managing ConvexOS:Operations Guide*.

## NAME

dump, vdump – incremental filesystem dump

## SYNOPSIS

*/etc/dump* key [ *argument ...* ] filesystem

## DESCRIPTION

*dump* copies all files changed after a certain date in the *filesystem* to magnetic tape. The *key* specifies the date and other options about the dump. *key* consists of characters from the set **0123456789afusdWnbGIE**.

- 0-9** This number is the “dump level”. All files modified since the last date stored in the file */etc/dumpdates* for the same filesystem at lesser levels will be dumped. If no date is determined by the level, the beginning of time is assumed; thus the option **0** causes the entire filesystem to be dumped.
- a** Makes *dump* skip over any files that have changed between the time that the *dump* was started and the time when *dump* actually copies the file. *dump* will not skip over directories and will print out a short message with the inode number of any files that were skipped. The **a** option is recommended when a mounted filesystem is *dumped*, since it will prevent invalid files from being written to the tape.
- f** Place the dump on the next *argument* file instead of the tape. If the name of the file is *-*, *dump* writes to standard output. The default is */dev/rmt8*.
- u** If the dump completes successfully, write the date of the beginning of the dump in */etc/dumpdates*. This file records a separate date for each filesystem and each dump level. The file */etc/dumpdates* is human readable. It consists of one free-format record per line: filesystem name, increment level, and *ctime(3)* format dump date. If necessary, */etc/dumpdates* may be edited to change any of the fields.
- s** The size of the dump tape is specified in feet. The number of feet is taken from the next *argument*. When the specified size is reached, *dump* will wait for reels to be changed. The default tape size is 2300 feet.
- d** The density of the tape, expressed in BPI, is taken from the next *argument*. This is used in calculating the amount of tape used per reel. The default is 1600.
- W** *dump* tells the operator what filesystems need to be dumped. This information is gleaned from the files */etc/dumpdates* and */etc/fstab*. The **W** option causes *dump* to print out, for each filesystem in */etc/dumpdates*, the most recent dump date and level, and highlights those filesystems that should be dumped. If the **W** option is set, all other options are ignored, and *dump* exits immediately.
- w** Is like **W**, but prints only those filesystems which need to be dumped.
- n** Whenever *dump* requires operator attention, notify all operators in the group “operator”, by means similar to a *wall(1)*.
- b** Take the next *argument* to be the blocking factor for tape records. This number indicates how many K-bytes are in a tape record. The default is 5.
- G** Set defaults for a GCR (6250 bpi) dump. This sets the tape unit to be */dev/rmt16*, the density to 6250, and the blocking factor to 50.
- I** Puts the tape drive into 100 ips streaming mode. The default is 50 ips.
- E** Causes *dump* to write to tape until finished or until end-of-tape is reached. If end-of-tape is reached, the operator is prompted to change tapes. Even with this option, *dump* will print an estimate of how many 9-track tapes (not cartridges) will be used for the dump.

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

Since making a full dump is time-consuming, *dump* checkpoints itself at the start of each tape volume. If writing that volume fails, *dump* will, with operator permission, restart itself from the checkpoint after the old tape has been rewound and removed, and a new tape has been mounted.

Since *dumps* are used to store potentially important data, filesystems should be unmounted when they are dumped. Mounted and active filesystems are the main causes of unreadable *dump* tapes. If it is not possible to unmount a filesystem, then *vdump* with the **a** option should be used.

*dump* tells the operator what is going on at periodic intervals, including estimates of the number of blocks to write, the number of tapes it will take, the time to completion, and the time to the tape change.

For each filesystem at your site, establish a sequence of full and incremental dumps on a weekly cycle. Because backing up and restoring a filesystem is a time-consuming process, the sequence you select has an impact on the amount of time required to perform backups and to restore files from dump tapes.

One possible backup schedule for performing dumps follows:

Day	Dump Level
Monday	level 0: backs up all files in the filesystem (full dump)
Tuesday	level 5: backs up all files changed since Monday's full dump
Wednesday	level 5: backs up all files changed since Monday's full dump
Thursday	level 5: backs up all files changed since Monday's full dump
Friday	level 5: backs up all files changed since Monday's full dump

With this method, restoring files requires only two sets of dump tapes: tapes for the full dump and tapes for the most recent incremental dump.

*vdump* performs the same function as *dump* but will also run *restore (8)* with the **V** option after the *dump* finishes. *restore (8)* in the verify mode checks the format of the *dump* tape and makes sure that the tape is readable. It is highly recommended that *vdump* with the **a** option be used for dumping filesystems that cannot be unmounted.

#### FILES

<i>/dev/rmt8</i>	default tape unit to dump to
<i>/etc/ddate</i>	old format dump date record (obsolete after <b>-J</b> option)
<i>/etc/dumpdates</i>	new format dump date record
<i>/etc/fstab</i>	dump table: filesystems and frequency
<i>/etc/group</i>	to find group <i>operator</i>

#### SEE ALSO

*mtio(4)*, *dump(5)*, *fstab(5)*, *restore(8)*, *xdump(8)*

#### DIAGNOSTICS

Unlike most other utilities, *dump* returns a 1 upon normal completion.

When dumping the */* directory it is not possible to unmount it. If the accounting are files changing, then the **a** option will cause them not be dumped.

#### BUGS

Sizes are based on 1600 bpi blocked tape; the raw magtape device has to be used to approach these densities. Fewer than 32 read errors on the filesystem are ignored (e.g., a warning message is issued). Greater than 32 read errors causes an abort. Each reel requires a new process, so

parent processes for reels already written, just hang around until the entire tape is written.

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

If a different tape blocking factor is used, it must also be used with *restore*.

**NAME**

dumpfs - dump file system information

**SYNOPSIS**

**dumpfs** *filesystem|device*

**DESCRIPTION**

*Dumpfs* prints out the super block and cylinder group information for the file system or special device specified. The listing is very long and detailed. This command is useful mostly for finding out certain file system information such as the file system block size and minimum free space percentage.

**SEE ALSO**

fs(5), disktab(5), newfs(8), fsck(8)

**NAME**

`edactwho` - edit `/etc/actwho` file

**SYNOPSIS**

**`edactwho`**

**`actwhocheck`** *actwhofile*

**DESCRIPTION**

*Edactwho* is a program to aid in editing the group-activities access control file `/etc/actwho`. *Edactwho* cooperates with the *bill* command to maintain a valid `/etc/actwho` file at all times. When *edactwho* is invoked, the user is taken to an editor to make changes in `/etc/actwho`. The editor the user is taken to is determined by the environment variable *EDITOR*; if *EDITOR* is not set, the editor *vi* is used. After editing is complete, the new file is checked for syntax, then moved to `/etc/actwho`.

*Actwhocheck* is the program which *edactwho* uses to validate the new `/etc/actwho` database; *actwhocheck* checks the syntax of the file given as its argument.

**FILES**

`/etc/actwho`

**SEE ALSO**

`bill(1)`, `actwho(5)`,  
"Accounting" chapter in the *CONVEX System Manager's Guide*.

**BUGS**

*Actwhocheck* only checks syntax, not content.

**NAME**

edquota - edit user quotas

**SYNOPSIS**

```
/usr/etc/edquota [-p proto-user] users...
/usr/etc/edquota -t
/usr/etc/edquota [-b bsoft] [-B bhard] [-i isoft] [-I ihard] filesystems users ...
```

**DESCRIPTION**

*edquota* is a quota editor. One or more users may be specified on the command line. If none of the batch mode flags are used (**b**, **B**, **i**, and/or **I**), then for each user a temporary file is created with an ASCII representation of the current disk quotas as well as the current disk usage for that user; an editor is then invoked on the file. The quotas may then be modified, new quotas added, etc. Upon leaving the editor, *edquota* reads the temporary file and modifies the binary quota files to reflect the changes made.

The editor invoked is *vi*(1) unless the EDITOR environment variable specifies otherwise.

Only the superuser may change or set quotas.

There is also a batch mode that will modify any or all of the four limits for a group of users for a single file system.

**OPTIONS**

- p duplicate the quotas of the prototypical user specified for each user specified. This is the normal mechanism used to initialize quotas for groups of users.
- t edit the soft time limits for each file system. If the time limits are zero, the default time limits in *<ufs/quota.h>* are used. Time units of sec(onds), min(utes), hour(s), day(s), week(s), and month(s) are understood. Time limits are printed in the greatest possible time unit such that the value is greater than or equal to one.
- b *blksoftlim* Set the block soft limit to the given number. This flag indicates batch mode (an editor is not invoked) and requires the additional argument *filesystems* before any of the users are listed.
- B *blkhardlim* Set the block hard limit to the given number. This flag indicates batch mode and requires the additional argument *filesystems* before any of the users are listed.
- i *inodesoftlim* Set the inode soft limit to the given number. This flag indicates batch mode and requires the additional argument *filesystems* before any of the users are listed.
- I *inodehardlim* Set the inode hard limit to the given number. This flag indicates batch mode and requires the additional argument *filesystems* before any of the users are listed.

**FILES**

<i>quotas</i>	quota file at the file system root
<i>/etc/mstab</i>	mounted file systems

**SEE ALSO**

quota(1), quotactl(2), quotacheck(8), quotaon(8), repquota(8)

**BUGS**

The format of the temporary file is inscrutable.

**NAME**

`faillogon` – log failed file access attempts

**SYNOPSIS**

`/etc/faillogon [ /usr/adm/failure_log ]`

**DESCRIPTION**

`faillogon` writes a log entry to `/usr/adm/failure_log` each time a process attempts to access a file for which it does not have permission. The file `/usr/adm/failure_log` must exist at the time you use `faillogon`. If you invoke `faillogon` with no argument, failure logging is turned off.

Failed file access logging is automatically disabled when the file system the log file resides on becomes 98% full; it is enabled again when the file system becomes less than 96% full. The percentages may be changed using `sysgen(8)`. See the “System Generation” chapter of the *CONVEX System Manager’s Guide* for more information.

**SEE ALSO**

`faillog(2)`, `faillogon(8)`, `faillogpr(8)`.

“System Protection” and “System Generation” chapters in the *CONVEX System Manager’s Guide*.

**BUGS**

In order to prevent possible file system deadlocks, failed attempts to access `/usr/adm/failure_log` are logged to the system console and the SPU file `/mnt/errlog` rather than to the specified log file.

## NAME

`faillogpr` – format the failed file access log

## SYNOPSIS

`/usr/adm/faillogpr [ logfiles ]`

## DESCRIPTION

`faillogpr` is a program that formats file access failure log files as produced by `faillogon(8)` in human-readable format. `faillogpr` takes the names of one or more log files, as created by `faillogon(8)` and described in `failure_log(5)`. If no arguments are specified, `faillogpr` reads its standard input.

`faillogpr`'s output is made up of one line of text for each log entry. The format is designed to be easily processed by `awk(1)`. Each line of text consists of each of the following fields separated by a single space character. There are no space characters in any of the fields except the *time* field.

*time uid errno mode command filename*

Each of the fields is described below.

*time* The time field shows the local time of the access. It has the format returned by `ctime(3)`. The field has five subfields: day of week, month, day of month, time of day, and year. Day of week is a three letter abbreviation, Sun, Mon, Tue, etc. Month is a three letter abbreviation, Jan, Feb, Mar, etc. Day of month is one or two digits. Time of day is of the form *hh:mm:ss* where *hh* is hours since midnight, *mm* is minutes past the hour, and *ss* is seconds past the minute.

*uid* This field contains the user ID's of the accessing program. If the program's real and effective user ID's are the same, the ID is only printed once. If they are different, i.e., the program is running set uid, the real ID is printed, followed by the effective ID in parentheses. Each ID is looked up in `/etc/passwd`. If a user is found who has that ID, the user's name is printed. Otherwise, the ID is printed as a decimal number.

*errno* This field contains the symbolic name of the error code that was returned from the system call that failed. The symbolic names are all listed in `/usr/include/errno.h` and also are listed and explained in `intro(2)`, as well as in the manual page for each system call.

*mode* The *mode* field is one or more of *R*, *W*, or *X*, indicating that the attempted access was for reading, writing, or execution, respectively. If the access was for none of those, it will be indicated as an underscore character, “\_”.

*command*

The name of the command is printed here. If the access is a Network File System (NFS) access from a remote machine, the command name will be printed as “nfsd”. If the command name contains non-printing or blank characters, those characters will be printed as `\ooo`, where *ooo* is the character's ASCII value in octal.

*filename*

The absolute pathname of the accessed file is printed. As with the command name, non-printing and blank characters are printed as octal escapes.

Note that `faillogpr` looks at the current state of the file systems to determine the names of files. The log file generated by the operating system only contains the device number and I-node number of the file. If the file has been deleted between the time when the failing access occurs and the time when `faillogpr` is run, the name will not be reported correctly, and the file name will be printed as `/aaa/bbb/ccc/<inumber>` where `/aaa/bbb/ccc` is the mount point of the file's file system and *inumber* is the file's I-node number. See `fs(5)` for an explanation of file systems, I-nodes and I-node numbers.

Also, `faillogpr` looks in `/etc/fstab` to determine where each file system is mounted. If the

information in */etc/fstab* is out of date, *faillogpr* may report pathnames incorrectly. When *faillogpr* can't find where a file system is mounted in */etc/fstab*, the first part of the pathname is printed as *<major/minor>*.

For example, a typical output line would look like this.

```
Wed May 6 21:15:19 1987 smith EACCES RW touch /etc/rc
```

This would indicate that at 9:15 pm local time on Wednesday, May 6th, 1987, the program *touch*, running with *smith's* user ID attempted to access the file */etc/rc* for both reading and writing and got the error return code *EACCES*.

#### NOTES

Because the file names generated by *faillogpr* are correct at the time when *faillogpr* is run, rather than the time when the failing access occurred, it is suggested that *faillogpr* should be run frequently so as to minimize the likelihood of incorrect file names being recorded. See the Log Files chapter of *Managing ConvexOS:Configuration Guide* for more information.

#### FILES

*/etc/fstab* */etc/ncheck* */etc/passwd*

#### SEE ALSO

*awk*(1), *intro*(2), *faillog*(2), *ctime*(3), *failure\_log* R(5), *fs*(5), *fstab*(5), *faillogon*(8).  
"Log Files" chapter in *Managing ConvexOS:Configuration Guide*.

#### BUGS

*faillogpr* reads its input twice. For that reason, it cannot take its input from a pipe or a socket.

**NAME**

*fastboot*, *fasthalt* – reboot/halt the system without checking the disks

**SYNOPSIS**

*/etc/fastboot* [ *boot-options* ]  
*/etc/fasthalt* [ *halt-options* ]

**DESCRIPTION**

*Fastboot* and *fasthalt* are shell scripts which reboot and halt the system without checking the file systems. This is done by creating a file */fastboot*, then invoking the *reboot* program. The system startup script, */etc/rc*, looks for this file and, if present, skips the normal invocation of *fsck*(8).

**SEE ALSO**

*halt*(8), *reboot*(8), *rc*(8)

**NAME**

*fingerd* – remote user information server

**SYNOPSIS**

*/usr/etc/in.fingerd*

**DESCRIPTION**

*fingerd* is a simple protocol based on RFC742 that provides an interface to the *name* and *finger* programs at several network sites. The program is supposed to return a friendly, human-oriented status report on either the system at the moment or a particular person in depth. There is no required format and the protocol consists mostly of specifying a single command line.

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

If the line is null (i.e., just a <CRLF> is sent), then *finger* returns a “default” report that lists all people logged into the system at that moment.

If a user name is specified (e.g., *eric*<CRLF>), then the response lists more extended information for only that particular user, whether logged in or not. Allowable names in the command line include both login names and user names. If a name is ambiguous, all possible derivations are returned.

**SEE ALSO**

*finger(1)*

**BUGS**

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

## NAME

*fsck* – file system consistency check and interactive repair

## SYNOPSIS

```
/etc/fsck -p [ filesystem ... ]
/etc/fsck [-b block# ] [-y ] [-n ] [-c ] [-d ] [ filesystem ] ...
```

## DESCRIPTION

The first form of *fsck* preens a standard set of filesystems or the specified file systems. It is normally used in the script */etc/rc* during automatic reboot. In this case *fsck* reads the table */etc/fstab* to determine which file systems to check. It uses the information there to inspect groups of disks in parallel taking maximum advantage of i/o overlap to check the file systems as quickly as possible. Normally, the root file system will be checked on pass 1, other “root” (“a” partition) file systems on pass 2, other small file systems on separate passes (e.g. the “d” file systems on pass 3 and the “e” file systems on pass 4), and finally the large user file systems on the last pass, e.g. pass 5. A pass number of 0 in *fstab* causes a disk to not be checked; similarly partitions which are not shown as to be mounted “rw” or “ro” are not checked.

The system takes care that only a restricted class of innocuous inconsistencies can happen unless hardware or software failures intervene. These are limited to the following:

- Unreferenced inodes
- Link counts in inodes too large
- Missing blocks in the free list
- Blocks in the free list also in files
- Counts in the super-block wrong

These are the only inconsistencies which *fsck* with the *-p* option will correct; if it encounters other inconsistencies, it exits with an abnormal return status and an automatic reboot will then fail. For each corrected inconsistency one or more lines will be printed identifying the file system on which the correction will take place, and the nature of the correction. After successfully correcting a file system, *fsck* will print the number of files on that file system and the number of used and free frags.

Without the *-p* option, *fsck* audits and interactively repairs inconsistent conditions for file systems. If the file system is inconsistent the operator is prompted for concurrence before each correction is attempted. It should be noted that a number of the corrective actions which are not fixable under the *-p* option 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*, *scheck*, and *icheck* combined.

The following flags are interpreted by *fsck*.

- b** Use the block specified immediately after the flag as the super block for the file system. Block 32 is always an alternate super block.
- y** Assume a yes response to all questions asked by *fsck*; this should be used with great caution as this is a free license to continue after essentially unlimited trouble has been encountered.
- n** Assume a no response to all questions asked by *fsck*; do not open the file system for writing.
- d** Certain diagnostic information is written as *fsck* goes about its work.
- c** Inode fields unused prior to version 9.0 of ConvexOS are zeroed; this flag should be use

when first upgrading to ConvexOS version 9.0. The following fields of the inode are cleared by this flag: *ic\_spare[0..3]*, *ic\_dmonflags*, *ic\_dmonperm*, and *ic\_sysflags*.

If no filesystems are given to *fsck* then a default list of file systems is read from the file */etc/fstab*.

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:
  - Directory size not of proper format.
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 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. The name assigned is the inode number. The only restriction is that the directory **lost+found** must preexist in the root of the filesystem being checked and must have empty slots in which entries can be made. This is accomplished by making **lost+found**, copying a number of files to the directory, and then removing them (before *fsck* is executed).

Checking the raw device is almost always faster.

#### FILES

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

#### DIAGNOSTICS

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

#### SEE ALSO

*fstab*(5), *fs*(5), *newfs*(8), *mkfs*(8), *reboot*(8)

#### BUGS

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

There should be some way to start a **fsck -p** at pass *n*.

**NAME**

*fsirand* - install random inode generation numbers

**SYNOPSIS**

***fsirand*** [ **-p** ] *special*

**DESCRIPTION**

*fsirand* installs random inode generation numbers on all the inodes on device *special*, and also installs a filesystem ID in the superblock. This helps increase the security of filesystems exported by NFS.

*fsirand* must be used only on an unmounted filesystem that has been checked with *fsck*(8). The only exception is that it can be used on the root filesystem in single-user mode, if the system is immediately re-booted afterwards.

**OPTIONS**

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

**NAME**

`fstat` - identify open files

**SYNOPSIS**

`fstat` [ `-u` user ] [ `-p` pid ] [ filename... ]

**DESCRIPTION**

*Fstat* identifies open files. A file is considered open if a process has it open or it is the working directory for a process. If no options are specified, *fstat* reports on all open files.

**OPTIONS**

`-u` requests a report of all files open by a specified user.

`-p` requests a report of all files open by a specified process id.

**filename...** restricts the reports to the specified files. If a file is a block special file, *fstat* additionally reports on any open files on that device, treating it as a mounted file system.

**OUTPUT**

The following fields are printed

**USER** is the user name of the owner of the process. If the user is not found in the password file, the id is printed out as its numeric equivalent.

**CMD** is the command name of the process.

**PID** is the process id.

**FD** is the file descriptor number in the per-process open file table. The special file descriptor `wd` signifies the working directory of the process.

If the file number is followed by an asterisk (\*), then the file is not an inode, but either a socket, process descriptor (see *pattach(2)*), a SPUio descriptor, or has an error of some kind. The format of the rest of the entry is variable, doesn't correspond to the headings, and is enclosed in parenthesis. The **SOCKET** section describes the variable format for sockets.

**DEVICE** is a display of the major and minor numbers of the device where this file's inode resides.

A device number of the form "255, XXX" shows the item in question to be remote mounted via NFS. The minor number, "XXX", is the number of that NFS mount, as found in */etc/mstab*. For example, "255, 10" means that the file in question is located on the 10th (starting at 0) NFS remote mount found in */etc/mstab*.

**INODE** is the inode number of the file.

**SIZE** is the size in bytes of the file, where applicable.

**TYPE** is the type of the file, as found in */usr/include/sys/vnode.h*

**FLAGS** There are the process flags as found in */usr/include/sys/vnode.h*. An `r` indicates that the file is the root of its file system. An `m` indicates that the file is a virtual memory object associated with the vnode.

**NAME** are the specified file names, if any.

**SOCKET**

The information displayed for an open socket depends on its protocol domain. The first three fields are always the same: field one is the domain name; field two is the socket type (stream, dgram, etc); and field three is the socket flags field in hex (see */usr/include/sys/socketvar.h*).

The remaining fields are protocol dependent. There is a final field for TCP sockets:

the address of the TCPCB; for UDP, the INPCB (socket PCB). For UNIX domain sockets, the remaining two fields display the address of the socket PCB and the address of the connected PCB. Otherwise, the protocol number and address of the socket itself are printed as the final field.

The intent is to supplement *netstat(1C)*. For example, the addresses mentioned above are the addresses that the “*netstat -A*” command would print for TCP, UDP, and UNIX domain.

Note that, since *pipe(2)* is implemented with sockets, a pipe appears as a connected UNIX domain stream socket. A unidirectional UNIX domain socket indicates the direction of flow with an arrow (“<-” or “->”), and a full duplex socket shows a double arrow (“<->”).

## EXAMPLES

This example shows the use of *fstat* to display information on the file */usr/spool/lpd/lock*.

```
% fstat /usr/spool/lpd/lock
```

USER	CMD	PID	FD	DEVICE	INODE	SIZE	TYPE	FLAGS	NAME
root	lpd	113	3	9, 35	26683	6	reg	x	/usr/spool/lpd/lock

This example shows the use of *fstat* to display all the files of a process that has */usr/spool/lpd/lock* open.

```
% fstat -p 113
```

USER	CMD	PID	FD	DEVICE	INODE	SIZE	TYPE	FLAGS	
root	lpd	113	wd	9, 35	26624	512	dir		
root	lpd	113	0	9, 0	4103	0	chr		
root	lpd	113	1	9, 0	4103	0	chr		
root	lpd	113	2	9, 35	26634	37	reg		
root	lpd	113	3	9, 35	26683	6	reg		
root	lpd	113	4*	(inet stream 80 tcp 80f6770c)					

This example shows the use of *netstat(1C)*, *grep(1)* and *fstat* to locate the process that has a socket open on the “smtp” network port. The *-aA* arguments direct *netstat* to display all protocol control block (PCB) addresses, and the *fstat* command displays all files, filtering the output through *grep* to locate the specific PCB.

```
% netstat -aA | grep smtp
```

```
80f67a8c tcp 0 0 *.smtp *.* LISTEN
```

```
% fstat | grep 80f67a8c
```

```
root sendmail 108 5* (inet stream 80 tcp 80f67a8c)
```

Finally, *fstat* can be used to help determine why a given filesystem (either locally or remotely mounted) cannot be unmounted. In this example, the local filesystem in question cannot be unmounted because both the regular file used by user john’s *vi* process, and the working directory for user bob’s *ls* is the root directory of the filesystem:

```
% umount /dev/du3a
```

```
/dev/du3a: Mount device busy
```

```
% fstat /dev/du3a
```

USER	CMD	PID	FD	DEVICE	INODE	SIZE	TYPE	FLAGS	NAME
john	vi	544	5	64, 769	21628	133	reg		/dev/du3a
bob	ls	698	wd	64, 769	2	752	dir	r	/dev/du3a

Since NFS mounts have no associated special block devices, one must use a different method to ferret out activity in them. As mentioned above, files on NFS mounts are given 255 as their major device number, and their position relative to other NFS mounts in */etc/mtab* as their minor number. Once this minor number is known, one can look for all NFS files open on that minor device number. In this example, john’s *vi* has a file open in *host2:/mrel*.

```
% umount /rmt/host2/mrel
/rmt/host2/mrel: Mount device busy
% grep : /etc/mtab
host0:/mnt /rmt/host0/mnt nfs bg,intr,timeo=30 0 0
host1:/usr/spool/globdata /rmt/host1/globdata nfs bg,intr,timeo=30 0 0
host2:/mrel /rmt/host2/mrel nfs bg,intr,timeo=30 0 0
% fstat | grep '255, *2'
john vi      3141 6      255, 2      822  9524 reg
```

*Fstat* is NOT useful in determining why a filesystem cannot be unmounted in the case of layered mounts. That is, *fstat* will not identify */usr/spool/mail* as the reason preventing */usr/spool* from being unmounted.

## BUGS

Socket information clutters the output.

*fstat* cannot find executing binaries which reside in a particular filesystem, but have no files open nor its working directory in that filesystem.

*fstat* cannot find inodes that are in use by the kernel, such as the inodes in use for the current accounting file (normally */usr/adm/acct*) and the current failure log file (normally */usr/adm/faillog*.)

Files being used as virtual memory objects, but which have been closed, are not identified.

Since *fstat* takes a snapshot of the system, it is only correct for a very short period of time.

## AUTHORS

*Fstat* comes from the 4.3BSD Tahoe distribution.

Vic Abell of the Purdue University Computing Center ported it to DYNIX 3.0.1[24] for the Sequent Balance and Symmetry machines, SunOS 4.0 and ULTRIX 2.2.

Tom Christiansen of Convex Computer Corporation ported it to the Convex, added the FLAGS field, the -m and -n options, the "process descriptor" and spuio types, decoded remote file descriptors, and improved the diagnostics on unknown user id's.

## SEE ALSO

ps(1), netstat(1C), pattach(2), pstat(8).

**NAME**

ftpd - DARPA Internet File Transfer Protocol server

**SYNOPSIS**

```
/etc/ftpd { -d } [ -l ] [ -t timeout ] [ -T maxtimeout ]
```

**DESCRIPTION**

*ftpd* is the DARPA Internet File Transfer Protocol (FTP) server process. The server uses the TCP protocol and listens at the port specified in the "ftp" service specification; see *services(5)*.

If the *-d* option is specified, debugging information is written to the *syslog*.

If the *-l* option is specified, each *ftp* session is logged in the *syslog*.

The *ftp* server will timeout an inactive session after 15 minutes. If the *-t* option is specified, the inactivity timeout period will be set to *timeout* seconds. A client may also request a different timeout period; the maximum period allowed may be set to *timeout* seconds with the *-T* option. The default limit is 2 hours.

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

Request	Description
ABOR	abort previous command
ACCT	specify account (ignored)
ALLO	allocate storage
APPE	append to a file
CDUP	change to parent of current working directory
CWD	change working directory
DELE	delete a file
HELP	give help information
LIST	give list files in a directory ( <i>ls -lgA</i> )
MKD	make a directory
MDTM	show last modification time of file
MODE	specify data transfer <i>mode</i>
NLST	give name list of files in directory
NOOP	do nothing
PASS	specify password
PASV	prepare for server-to-server transfer
PORT	specify data connection port
PWD	print the current working directory
QUIT	terminate session
RETR	retrieve a file
RMD	remove a directory
RNFR	specify rename-from file name
RNTO	specify rename-to file name
SITE	non-standard commands (see next section)
SIZE	return size of file
STAT	return status of server
STOR	store a file
STOU	store a file with a unique name
STRU	specify data transfer structure
SYST	show operating system type of server system
TYPE	specify data transfer type
USER	specify user name
XCUP	change to parent of current working directory (deprecated)
XCWD	change working directory (deprecated)
XMKD	make a directory (deprecated)
XPWD	print the current working directory (deprecated)

**XRMD** remove a directory (deprecated)

The following nonstandard or UNIX-specific commands are supported by the SITE request:

Request	Description
UMASK	change umask, e.g., <b>SITE UMASK 002</b>
IDLE	set idle-timer, e.g., <b>SITE IDLE 60</b>
CHMOD	change mode of a file, e.g., <b>SITE CHMOD 755 filename</b>
HELP	give help information, e.g., <b>SITE HELP</b>

The remaining *ftp* requests specified in Internet RFC 959 are recognized, but not implemented. MDTM and SIZE are not specified in RFC 959, but will appear in the next updated FTP RFC.

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

*ftpd* interprets file names according to the "globbing" conventions used by *cs(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) The user must have a standard shell returned by *getusershell(3)*.
- 4) If the user name is "anonymous" or "ftp", an anonymous *ftp* account must be present in the password file (user "ftp"). In this case, the user is allowed to log in by specifying any password (by convention, this is given as the client host's name).

In the last case, *ftpd* takes special measures to restrict the client's access privileges. The server performs a *chroot(2)* command to the home directory of the *ftp* user. So 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 not writable by anyone.

~ftp/bin)

Make this directory owned by the super-user and not writable by anyone. The program *ls(1)* must be present to support the *list* command. This program should have mode 111.

~ftp/etc)

Make this directory owned by the super-user and not writable by anyone. The files *passwd(5)* and *group(5)* must be present for the *ls* command to be able to produce owner names rather than numbers. The password field in *passwd* is not used, and should not contain real encrypted passwords. These files should be mode 444.

~ftp/pub)

Make this directory mode 777 and owned by "ftp." Users should then place files that are to be accessible via the anonymous account into this directory.

#### SEE ALSO

*ftp(1)*, *getusershell(3)*, *syslogd(8)*

#### BUGS

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 the list of known holes is possibly incomplete.

**NAME**

genrest - generate password restrictions file

**SYNOPSIS**

**genrest** [ *-t* ] [ *-m**n* ] [ *-M**N* ]

**DESCRIPTION**

*Genrest* creates the password restrictions file */etc/pwrestrict* from the password file */etc/passwd*. Each line in */etc/passwd* is scanned, and the requisite fields are extracted to make an entry for the restrictions file. Finally, *mkpasswd(8)* is used to build the database files */etc/pwrestrict.dir* and */etc/pwrestrict.pag*.

By default, no restriction are enabled when */etc/pwrestrict* is built. If the *-t* option is specified, type restrictions will be enabled for every user. If either the *-m* or *-M* option is given password aging will be enabled; the *-m* option sets the minimum period in weeks which must lapse before the password can be changed to *n*. The *-M* option sets the maximum number of weeks for which a password is valid to *N*. The default minimum is 1 week; the default maximum is 52 weeks.

To set user-specific password restrictions, edit */etc/pwrestrict* after *genrest* using the *vipw(8)* utility.

Genrest will not build the */etc/pwrestrict* file if it already exists.

The file */etc/pwrestrict* is *optional* and may be omitted; it is only needed by sites desiring password restrictions.

**SEE ALSO**

*passwd(5)*, *pwrestrict(5)*, *mkpasswd(8)*, *vipw(8)*

**NAME**

getty - set terminal mode

**SYNOPSIS**

*/etc/getty* [ type ]

**DESCRIPTION**

*Getty* is invoked by *init(8)* immediately after a terminal is opened, following the making of a connection. While reading the name *getty* attempts to adapt the system to the speed and type of terminal being used.

*Init* calls *getty* with an argument specified by the *ttys* file entry for the terminal line. The argument can be used to make *getty* treat the line specially. This argument is used as an index into the *gettytab(5)* database, to determine the characteristics of the line. If there is no argument, or there is no such table, the **default** table is used. If there is no */etc/gettytab* a set of system defaults is used. If indicated by the table located, *getty* will clear the terminal screen, print a banner heading, and prompt for a login name. Usually either the banner or the login prompt will include the system hostname. Then the user's name is read, a character at a time. If a null character is received, it is assumed to be the result of the user pushing the 'break' ('interrupt') key. The speed is usually then changed and the 'login:' is typed again; a second 'break' changes the speed again and the 'login:' is typed once more. Successive 'break' characters cycle through some standard set of speeds.

The user's name is terminated by a newline or carriage-return character. The latter results in the system being set to treat carriage returns appropriately (see *tty(4)*).

The user's name is scanned to see if it contains any lowercase alphabetic characters; if not, and if the name is nonempty, the system is told to map any future uppercase characters into the corresponding lowercase characters.

Finally, login is called with the user's name as argument.

Most of the default actions of *getty* can be circumvented, or modified, by a suitable *gettytab* table.

*Getty* can be set to timeout after some interval, which will cause dial-up lines to hang up if the login name is not entered reasonably quickly.

**FILES**

*/etc/gettytab*

**SEE ALSO**

*gettytab(5)*, *init(8)*, *login(1)*, *ioctl(2)*, *tty(4)*, *ttys(5)*.

**NAME**

`getst` - print information from the kernel's stripe table

**SYNOPSIS**

`getst /dev/rst? ...`

**DESCRIPTION**

`getst` retrieves stripe table information from the kernel for each striped file system given as an argument and prints a brief summary of each to *stdout*. For each stripe device argument, information such as the partitions which make up the stripe, the number of blocks used on each partition, and the stripe blocking factor will be displayed. You will have to be superuser to read the stripe table from the kernel.

See `st(4)` or `putst(8)` for more information on stripes.

**SEE ALSO**

`st(4)`, `newst(8)`, `putst(8)`

**NAME**

gettable - get NIC format host tables

**SYNOPSIS**

```
/etc/gettable [ -v ] host [ outfile ]
```

**DESCRIPTION**

*gettable* is a simple program used to obtain the NIC standard host tables from a "nickname" server. The indicated *host* is queried for the tables. The tables, if retrieved, are placed in the file, *outfile*, or by default, *hosts.txt*.

The *-v* option will get just the version number instead of the complete host table and put the output in the file, *outfile*, or by default, *hosts.ver*.

*gettable* operates by opening a TCP connection to the port indicated in the service specification for "nickname." A request is then made for "ALL" names and the resultant information is placed in the output file.

*gettable* is best used in conjunction with the *htable(8C)* program, which converts the NIC standard file format to that used by the network library lookup routines.

**SEE ALSO**

intro(3), *htable(8C)*, *named(8)*

**BUGS**

If the name server system provided network name mapping well as host name mapping, *gettable* would no longer be needed.

**NAME**

halt - stop the processor

**SYNOPSIS**

`/etc/halt [-n] [-q] [-y]`

**DESCRIPTION**

*Halt* writes out sandbagged information to the disks and then stops the processor. The machine does not reboot, even if the auto-reboot switch is set on the console.

The `-n` option prevents the sync before stopping. The `-q` option causes a quick halt, no graceful shutdown is attempted. The `-y` option is needed if you are trying to halt the system from a dialup.

**SEE ALSO**

reboot(8), shutdown(8)

## NAME

`hconfig` – Maintains the HASP Host communications capability database

## SYNOPSIS

`hconfig` [ `-inr` ]

## DESCRIPTION

`hconfig` creates, maintains, saves, and restores the HASP Host communications capability database, `commcap(5)`.

The switches have the following meaning:

`-i` initializes the `commcap(5)` file. This is a very destructive switch. It should be used only once, when the product is initially loaded. To protect against accidental use, the question, “*WARNING -- You are about to initialize a key HASP Host file. Are you sure? (Y/N)*” will be displayed. Type **Y** to perform the initialization or **N** to abort it.

`-n` adds new entries to the `commcap(5)` file. The editor chosen by the environment variable `EDITOR` (see `cs(1)`) is invoked with a template for you to fill in. This template contains default `commcap(5)` fields and values. See `commcap(5)` for the meaning of an entry.

`-r` restores a backup version of the `commcap(5)` file.

If there are no switches on the command line then the `commcap(5)` file will be edited and validated. If you add a switch, choose only one.

`hconfig` works by using the editor of your choice on the `commcap(5)` file. When you exit your editor, `hconfig` then scans the file, validating all entries. If a problem is found with an entry, a message is displayed on your screen telling you what was found. The program will wait about 5 seconds so that you may read the message. It will then reenter your editor. When the validation is successful, `hconfig` will exit.

Aborting your editor will return to `hconfig`. Aborting `hconfig` while it is validating entries may be done but is unwise since the consistency of the HASP Host capabilities becomes uncertain.

`hconfig` saves a copy of `commcap(5)` to `/usr/hasp/etc/commcap.bak` before you begin editing. If you initialize the database (with the `-i` switch), a copy of `commcap(5)` is NOT saved.

## VARIABLES

## EDITOR

Set to the editor of choice for making changes to options. If this variable is missing, `vi(1)` is used. If the contents of this variable are erroneous, `hconfig` fails.

## FILES

<code>/usr/hasp/etc/commcap</code>	The configuration options file ( <code>commcap(5)</code> )
<code>/usr/hasp/etc/commcap.bak</code>	Copy of the communications capability database
<code>/tmp/HASPxxxxxx</code>	Temporary files for massaging options

## DIAGNOSTICS

A number of messages may be displayed by `hconfig`. These are usually due to a mistake of some sort. Consult the following list to identify and resolve the problem.

Are you sure? (Y/N)

This is a safety-valve question when you specify `-i` on the command line. If you are certain about initializing the `commcap(5)` database, type **Y**. Otherwise, type **N**.

Backup file `filename` can't be opened

During a restore operation, the backup file, `filename`, couldn't be opened. The restore operation is aborted. A second message appears after this message explaining the reason why. Satisfy the message and try again.

**Can't add new entry**

The new *commcap*(5) entry that was just edited cannot be added to the *commcap*(5) file. A second message appears after this message explaining the reason why. Satisfy the message and try again.

**Can't backup *filename***

Making a copy of *filename* was unsuccessful. A second message appears after this message explaining the reason why. Satisfy the message and try again.

**Can't copy *filename1* to *filename2***

During a restore operation, the backup file, *filename1*, couldn't be copied to *filename2*. The restore operation is aborted. A second message appears after this message explaining the reason why. Satisfy the message and try again.

**Can't create *filename***

During an initialize operation, *filename* couldn't be created. The initialize operation is aborted. A second message appears after this message explaining the reason why. Satisfy the message and try again.

**Can't create temporary file *filename***

While preparing for a new *commcap*(5) entry, the temporary file *filename* couldn't be opened. A second message appears after this message explaining the reason why. Satisfy the message and try again.

**Can't edit new entry**

The editor that you have chosen with the *EDITOR* environment variable cannot be invoked. The *commcap*(5) file cannot be changed. A second message appears after this message explaining the reason why. Satisfy the message or insure that *EDITOR* points to a valid editor. In either case, try again.

**Can't read *filename***

The *commcap*(5) file cannot be read for validation. Verify that it exists and that read permission is granted, then run *hconfig* again.

**Configuration utility failed**

The editor that you have chosen with the *EDITOR* environment variable cannot be invoked. The *commcap*(5) file cannot be changed. A second message appears after this message explaining the reason why. Satisfy the message or insure that *EDITOR* points to a valid editor. In either case, try again.

**Entry *name* has an illegal value of *string* for *field***

The *commcap*(5) entry with the remote workstation name of *name* has a mistake. The *field* field has a value of *string* which is incorrect. See *commcap*(5) for the proper values and correct the error. You will be presented the file again in order to correct the mistake.

**Entry *name* is missing field *field***

The *commcap*(5) entry with the workstation name of *name* has a mistake. The *field* field is missing. See *commcap*(5) for the meaning of this field and the proper values. You will be presented the file again in order to correct the mistake.

**Too many switches on the command line.**

There can be only one switch specified on the command line. Re-enter the command with the appropriate one.

**WARNING -- You are about to initialize a key HASP Host file.**

This is a warning that is displayed when you include the *-i* switch on the command line. A question follows this message, forcing you to verify that you want to initialize the *commcap*(5) database.

usage: `hconfig [-inr]`

If a mistake is made when invoking *hconfig*, you may receive this as a reminder of the proper way to execute the program.

*filename* validation failed

The validation process detected an error for file, *filename*. Correct the mistake that is noted in a preceding message. You will be presented the file again in order to do so.

#### SEE ALSO

`commcap(5)`, `hsend(8)`

#### NOTES

Using the `-i` switch does not backup the `commcap(5)` file. It also doesn't permit a general editing session after its creation. You must invoke *hconfig* again to add entries or to edit the file.

*hconfig* is part of the CONVEX HASP Host Communications Product. HASP Host is an optional product; for more information, contact your CONVEX sales representative.

## NAME

hhstart - Start HASP Host

## SYNOPSIS

**hhstart** [*board\_number*]

## DESCRIPTION

*hhstart* is a *cs*(1) script that starts the HASP Host software on communications boards.

*board\_number* is the number of a communications board where HASP Host is started. Board numbers begin with 0. If omitted, HASP Host is started on all communications boards contained in the *current\_rmts*(5) database.

## FILES

/usr/hasp/bin/hhstart	This script
/usr/hasp/etc/current_rmts	HASP Host board database

## SEE ALSO

*current\_rmts*(5), *hhstop*(8), *seth*(8)

## NOTES

*hhstart* is part of the CONVEX HASP Host Communications Product. HASP Host is an optional product; for more information, contact your CONVEX sales representative.

**NAME**

hhstop – Stop HASP Host

**SYNOPSIS**

**hhstop** [*board\_number*]

**DESCRIPTION**

*hhstop* is a *ckh*(1) script that stops the HASP Host software on communications boards.

*board\_number* is the number of a communications board where HASP Host is stopped. Board numbers begin with 0. If omitted, HASP Host is stopped on all communications boards contained in the *current\_rmts*(5) database.

**FILES**

<i>/usr/hasp/bin/hhstop</i>	This script
<i>/usr/hasp/etc/current_rmts</i>	HASP Host board database

**SEE ALSO**

*current\_rmts*(5), *hhstart*(8), *sethbb*(8)

**NOTES**

*hhstop* is part of the CONVEX HASP Host Communications Product. HASP Host is an optional product; for more information, contact your CONVEX sales representative.

## NAME

**hsend** - Transfer file from a Convex computer to a remote workstation

## SYNOPSIS

**hsend** [-d] [-j *jobname*] [-r *remote*] [-s *stream*] *filename*

## DESCRIPTION

*hsend* is a utility that is part of the interface software used by the HASP Host Communications Product. Its purpose is to send jobs from a Convex computer to a remote workstation. This utility is generally used for returning output files from a batch job. It may be used by an interactive user on a Convex computer, provided that some caution is exercised.

The switches and parameters have the following meaning:

-d deletes *filename* on the Convex computer after the transfer is successful. This switch should not be used by an interactive user.

-j *jobname* is the job identifier associated with the output. It is used by other components of the interface software. This switch should not be used by an interactive user.

-r *remote* is the name of the remote workstation to receive the file. If this parameter is omitted, the remote name "edmund" will be used.

-s *stream* is the name of the output channel where *filename* is sent on the remote workstation. If this parameter is omitted, the stream name "prt1" will be used.

*filename* is the name of the file to send to the remote workstation. If omitted, this command will fail.

## EXAMPLES

**hsend -d -r rook castle.c**

The file *castle.c* is transferred to the *prt1* stream on the remote workstation, *rook*. After *rook* is transferred, it will be deleted.

**hsend -r calgary -s pun1 /tmp/well.log**

The file */tmp/well.log* is transferred to the *pun1* stream on the remote workstation, *calgary*. The file is not deleted after the transfer.

## DIAGNOSTICS

*filename* doesn't exist

The supplied *filename* doesn't exist. This is caused by a misspelling or an incorrect directory path. Correct the *filename* and reenter the command.

Internal error (popen)

Cause is '*message text*'

*hsend* encountered a problem while communicating with other components of the HASP Host Product. The *message text* describes the error in greater detail. This is a fatal error. Your file was not transferred. Contact your system manager if this occurs.

Missing *argument*

A switch was placed on the command line but the associated *argument* was omitted. For example, the -r switch was given but the *remote* name wasn't. Correct the parameters and reenter the command.

Too few parameters

Not enough parameters were on the command line. *hsend* expects a minimum pattern of **hsend *filename***. Reenter the command with more parameters.

usage: **hsend** [-d] [-j *jobname*] [-r *remote*] [-s *stream*] *filename*

If you made a mistake when entering the *hsend* command, you may receive this display following an error message. It is a reminder of how to use this command.

**STATUS**

0 is returned if the command was successful. Otherwise, 1 is returned.

**SEE ALSO**

commcap(5)

**BUGS**

This program assumes a sophisticated user. There is no way to validate the *remote* and *stream* parameters. If they are incorrect, then the file will not reach its destination.

**NOTES**

The interactive user must obtain the *remote* and *stream* values from the system manager.

*hsend* is part of the CONVEX HASP Host Communications Product. HASP Host is an optional product; for more information, contact your CONVEX sales representative.

**NAME**

*htable* - convert NIC standard format host tables

**SYNOPSIS**

*/etc/htable* [ *-c connected-nets* ] [ *-l local-nets* ] *file*

**DESCRIPTION**

*htable* is used to convert host files in the format specified in Internet RFC 810 to the format used by the network library routines. Three files are created as a result of running *htable*: *hosts*, *networks*, and *gateways*. The *hosts* file may be used by the *gethostbyname(3N)* routines in mapping host names to addresses if the nameserver, *named(8)*, is not used. The *networks* file is used by the *getnetent(3N)* routines in mapping network names to numbers. The *gateways* file may be used by the routing daemon in identifying "passive" Internet gateways; see *routed(8C)* for an explanation.

If any of the files, *localhosts*, *localnetworks*, or *localgateways*, is present in the current directory, the file's contents is prepended to the output file. Of these, only the *gateways* file is interpreted. This allows sites to maintain local aliases and entries which are not normally present in the master database. Only one gateway to each network will be placed in the *gateways* file; a gateway listed in the *localgateways* file will override any in the input file.

If the *gateways* file is to be used, a list of networks to which the host is directly connected is specified with the *-c* flag. The networks, separated by commas, may be given by name or in dot notation, e.g., *-c arpanet,128.32,local-ether-net*. *htable* includes only gateways that are directly connected to one of the networks specified, or that can be reached from another gateway on a connected network.

If the *-l* option is given with a list of networks (in the same format as for *-c*), these networks will be treated as "local," and information about hosts on local networks is taken only from the *localhosts* file. Entries for local hosts from the main database will be omitted. This allows the *localhosts* file to completely override any entries in the input file.

*htable* is best used in conjunction with the *gettable(8C)* program, which retrieves the NIC database from a host.

**SEE ALSO**

*intro(4n)*, *gettable(8C)*, *named(8)*

**BUGS**

If the name server system provided network name mapping well as host name mapping, *htable* would no longer be needed.

**NAME**

hyroute - set the HYPERchannel routing tables

**SYNOPSIS**

```
hyroute interface [-s] [-p] [-c] [file]
hyroute interface [-d]
```

**DESCRIPTION**

*hyroute* manipulates the HYPERchannel routing information.

The *interface* parameter specifies the HYPERchannel interface name as a string of the form "name unit," e.g., "hy0." With the *-s* option, it reads *file* and sets the system's database according to the information in the file (see below). If no input file is given, or if the argument "-" is encountered, *hyroute* reads from standard input.

The *-c* option causes *hyroute* to compare the system's current information to that contained in *file*.

The *-p* option causes a digested version of *file* to be printed.

The *-d* option causes a "dump" of the system's table (used for debugging routing code).

**FILE FORMAT**

The input file is free format. Comment lines start with an asterisk in column one. Statements end with a semicolon. The command:

```
direct host dest control access MTU;
```

describes a host that can be reached directly from this adapter. *host* is a host name as listed in */etc/hosts*; *dest*, *control*, *access*, and *MTU* are hexadecimal numbers. Data written to a given host will be sent to HYPERchannel address *dest* using a control value of *control*, an access code of *access* (see adapter manuals for details), and fragmented such that the maximum transmission unit is (*MTU* times 1024) plus 54 bytes of user data available in a HYPERchannel message proper. Valid MTUs range from 1 Kbytes to 32 Kbytes. The default MTU is 4 Kbytes.

The command:

```
gateway host gate1 gate2 gate3 ... ;
```

describes a host that must be reached indirectly through any one of the designated hosts. Listed hosts are not gateways in the formal sense (they do not run the internet gateway protocols), but are hosts on the HYPERchannel that can "bridge" between subsections of the HYPERchannel network.

A sample file follows:

```
* comment
direct azure    6100 0 0 4;
direct bronze   6101 0 0 4;
direct cyber    2100 1100 0 16;
direct dadcad   6102 0 0 4;
direct tekcad   2400 1100 0 16;
direct tekcrd   2201 1100 0 16;
direct tekid    2500 1100 0 16;
direct teklabs  2200 1100 0 16;
gateway iddic  tekcrd teklabs cyber tekcad tekid;
gateway iddme  tekcrd teklabs cyber tekcad tekid;
gateway metals tekcrd teklabs cyber;
```

**SEE ALSO**

hy(4), hystat(8c)

**NAME**

`hystat` – acquire and display HYPERchannel adapter statistics

**SYNOPSIS**

`hystat interface [-d ] [-t interval ] [-r repeat ]`

**DESCRIPTION**

`hystat` requests a given HYPERchannel *interface* to return its statistics and status.

The following describes flags and options:

If `-r` is given, the request and display are repeated the number of times given in the *repeat* count.

If `-t` is given, the interval between repeats is the number of seconds given by *interval*.

If `-d` is given, the display will be of the deltas of counters returned between repeats in a given execution of `hystat`. This display will scroll to *stdout*. Otherwise, information is displayed cumulatively on a static screen template.

Information given on the scrolled display is per trunk; for each of the four adapter trunks, accumulators of the message count and the retransmit count are displayed. In addition, an abort count and cancel count global to the adapter are displayed.

The static display shows these values as well as the adapter type, status byte, last function, trunk response, trunk status, received response, error code, and first message index.

**SEE ALSO**

`netstat(1C)`

**BUGS**

Trunk statistics are valid for the NSC A400 adapter. They may not be visible to the NSC NB400 adapter.

**NAME**

`icheck` - file system storage consistency check

**SYNOPSIS**

`/etc/icheck` [ `-s` ] [ `-b` numbers ] [ filesystem ]

**DESCRIPTION**

**N.B.:** *icheck* is obsoleted for normal consistency checking by *fsck*(8).

*icheck* examines a file system, builds a bit map of used blocks, and compares this bit map against the free list maintained on the file system. If the file system is not specified, a set of default file systems is checked. The normal output of *icheck* includes a report of

The total number of files and the numbers of regular, directory, block special and character special files.

The total number of blocks in use and the numbers of single-, double-, and triple-indirect blocks and directory blocks.

The number of free blocks.

The number of blocks missing; i.e. not in any file nor in the free list.

The `-s` option causes *icheck* to ignore the actual free list and reconstruct a new one by rewriting the super-block of the file system. The file system should be dismounted while this is done; if this is not possible (for example if the root file system has to be salvaged) care should be taken that the system is quiescent and that it is rebooted immediately afterwards so that the old, bad in-core copy of the super-block will not continue to be used. Notice also that the words in the super-block which indicate the size of the free list and of the i-list are believed. If the super-block has been curdled these words will have to be patched. The `-s` option causes the normal output reports to be suppressed.

Following the `-b` option is a list of block numbers; whenever any of the named blocks turns up in a file, a diagnostic is produced.

*icheck* is faster if the raw version of the special file is used, since it reads the i-list many blocks at a time.

**FILES**

Default file systems vary with installation.

**SEE ALSO**

*fsck*(8), *dcheck*(8), *ncheck*(8), *fs*(5), *clri*(8)

**DIAGNOSTICS**

For duplicate blocks and bad blocks (which lie outside the file system) *icheck* announces the difficulty, the i-number, and the kind of block involved. If a read error is encountered, the block number of the bad block is printed and *icheck* considers it to contain 0. 'Bad freeblock' means that a block number outside the available space was encountered in the free list. 'n dups in free' means that *n* blocks were found in the free list which duplicate blocks either in some file or in the earlier part of the free list.

**BUGS**

Since *icheck* is inherently two-pass in nature, extraneous diagnostics may be produced if applied to active file systems.

It believes even preposterous super-blocks and consequently can get core images.

The system should be fixed so that the reboot after fixing the root file system is not necessary.

## NAME

`ifconfig` – configure network interface parameters

## SYNOPSIS

```
/etc/ifconfig interface [ address [ dest_address ] ] [ parameters ]
/etc/ifconfig interface [ protocol_family ]
```

## 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 or other operating parameters. The `interface` parameter is a string of the form "name unit", e.g., `ex0`. The address is either a host name present in the host name data base, `hosts(5)`, or a DARPA Internet address expressed in the Internet standard "dot notation."

The following parameters may be set with `ifconfig`:

- up** Mark an interface "up." This may be used to enable an interface after an "ifconfig down." It happens automatically when setting the first address on an interface. If the interface was reset when previously marked down, the hardware will be re-initialized.
- down** Mark an interface "down." When an interface is marked "down," the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.
- trailers** Request the use of a "trailer" link level encapsulation when sending (default). If a network interface supports `trailers`, the system will, when possible, encapsulate outgoing messages in a manner that minimizes the number of memory-to-memory copy operations performed by the receiver. On networks that support the Address Resolution Protocol (see `arp(4P)`); currently, only 10 Mb/s Ethernet, this flag indicates that the system should request that other systems use trailers when sending to this host. Similarly, trailer encapsulations will be sent to other hosts that have made such requests. Currently used by Internet protocols only.
- trailers** Disable the use of a "trailer" link level encapsulation.
- arp** Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between DARPA Internet addresses and 10Mb/s Ethernet addresses.
- arp** Disable the use of the Address Resolution Protocol.
- metric *n*** Set the routing metric of the interface to *n*, default 0. The routing metric is used by the routing protocol (`routed(8c)`). Higher metrics have the effect of making a route less favorable; metrics are counted as addition hops to the destination network or host.
- debug** Enable driver dependent debugging code; usually, this turns on extra console error logging.
- debug** Disable driver dependent debugging code.
- netmask *mask*** (Inet only) Specify how much of the address to reserve for subdividing networks into subnetworks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation Internet address, or with a pseudo-network name listed in the network table, `networks(5)`. The mask contains 1's for the bit positions in the 32-bit

address that are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.

**dest\_address** Specify the address of the correspondent on the other end of a point-to-point link.

**broadcast** (Inet only) Specify the address to use to represent broadcasts to the network. The default broadcast address is an address with a host part of all 1's.

*ifconfig* displays the current configuration for a network interface when no optional parameters are supplied. If a protocol family is specified, *ifconfig* will report only the details specific to that protocol family.

Only the super-user may modify the configuration of a network interface.

#### DIAGNOSTICS

Messages indicating the specified interface does not exist, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

#### SEE ALSO

netstat(1C), intro(4n), rc(8), ex(4)

## NAME

inetd, inetd.conf – internet super-server daemon and configuration file

## SYNOPSIS

`/etc/inetd [ -d ] [ configuration_file ]`

## DESCRIPTION

*inetd* is normally run at boot time by the `/etc/rc` startup file. It then listens for connections on certain internet sockets. When a connection is found on one of its sockets, it decides what service the socket corresponds to, and invokes a program to service the request. This process is passed the connection as file descriptor 0 and an argument of the form "sourcehost.sourceport" where "sourcehost" is hex and "sourceport" is decimal, as well as any arguments specified in the configuration file. After the program is finished, it continues to listen on the socket (except in some cases, as described below). Essentially, *inetd* allows you to run one daemon to invoke several others, thus reducing load on the system.

Upon execution, *inetd* reads its configuration information from a file that, by default, is `/etc/inetd.conf`. There must be an entry for each field of the configuration file; fields are separated by tabs or spaces. Comments are denoted by a "#" at the beginning of a line. The fields of the configuration file are as follows:

```

service name
socket type
protocol
wait/nowait
user
server program
server program arguments

```

*service name* is the name of a valid service in the file `/etc/services` or `/etc/rpc`, if the entry is for an RPC program (i.e., a version number is given. See *server program*, below). For "internal" services (discussed below), the service name must be the official name of the service (that is, the first entry in `/etc/services`).

*socket type* is one of "stream," "dgram," or "raw," depending on whether the socket handles stream, datagram, or raw data. Most RPC programs use the "dgram" socket type.

*protocol* must be a valid protocol as listed in `/etc/protocols`. Examples might be "tcp" or "udp." In general, if the *socket type* is specified as "stream," then this entry is "tcp." Most RPC programs use the "udp" protocol.

*wait/nowait* is applicable only to datagram sockets; other sockets should have a "nowait" entry in this position. If a datagram server connects to its peer, freeing the socket so *inetd* can receive further messages on the socket, it is said to be a "multi-threaded" server, and should use the "nowait" entry. For datagram servers that process all incoming datagrams on a socket and eventually time out, the server is said to be "single-threaded," and should use a "wait" entry. *comsat*(8C), *biff*(1), and *talk*(8C) are all examples of the latter type of datagram server. An exception is *tftpd*; it is a datagram server that establishes pseudo-connections. It must be specified as "wait" in order to avoid a race condition; the server reads the first packet, creates a new socket, and then forks and exits to allow *inetd* to check for new service requests to spawn new servers. Most RPC programs use the "wait" entry.

*user* contains the username of the user as whom the server should run. This allows you to give servers less permission than root.

*server program* contains the pathname of the program *inetd* should execute when a request is found on its socket. If this is an RPC (Remote Procedure Call) daemon, this entry first specifies the version number(s) supported by the program, followed by a space, then the pathname of the program to execute. The version number may be a single digit or a range specified as two digits separated by "-", as in "1-4." If *inetd* provides this service internally, this entry should be

“internal.”

*server program arguments* are specified just as you normally specify program arguments, starting with `argv[0]`, which is the name of the program. If the service is provided internally, this entry is not used and need not be specified. Up to 8 additional arguments may be specified; they are passed to the program normally.

*inetd* provides several “trivial” services through the use of internal routines. These services are “echo,” “discard,” “chargen” (character generator), “daytime” (human-readable time), and “time” (machine-readable time represented as the number of seconds since midnight, January 1, 1900). All of these services are TCP based. For details on these services, consult the appropriate RFC from the Network Information Center.

*inetd* rereads its configuration file when it receives a hangup signal, *SIGHUP*. Services may be added, deleted, or modified when the configuration file is reread.

## RPC SUPPORT

An RPC server can be started from *inetd*. The only differences from the usual code are that *svcmdp\_create* is called as:

```
transp == svcmdp_create(0)
```

because *inetd* passes a socket file as descriptor 0; and *svc\_register* is called as:

```
svc_register(PROGNUM, VERSNUM, service, transp, 0)
```

with the final flag as 0, because the program will already have been registered by *inetd*. If you want to exit from the server process and return control to *inetd*, you must explicitly exit, because *svc\_run* never returns.

## OPTIONS

**-d** Specifies that debugging traces are to be turned on for connection-oriented (TCP) services.

### configuration\_file

Specifies an alternate configuration file to use instead of */etc/inetd.conf*.

## EXAMPLE

The following is an example */etc/inetd.conf* configuration file:

```
ftp      stream tcp nowait root /usr/etc/in.ftpd      ftpd -l/tmp/ftp.log
telnet   stream tcp nowait root /usr/etc/in.telnetd    telnetd
shell    stream tcp nowait root /etc/in.rshd           rshd
login    stream tcp nowait root /etc/in.rlogind        rlogind -d
exec     stream tcp nowait root /usr/etc/in.rexecd       rexecd
syslog   dgram  udp wait  root /usr/etc/in.syslog     syslog -t5
comsat   dgram  udp wait  root /usr/etc/in.comsat    comsat
talk     dgram  udp wait  root /usr/etc/in.talkd     talkd
crashdump dgram  udp wait  root /usr/etc/in.crashreceive crashreceive
rusers   dgram  udp wait  root 1-2 /usr/etc/rpc.rusersd rusersd
rup      dgram  udp wait  root 1-3 /usr/etc/rpc.rstatd rstatd
rwall    dgram  udp wait  root 1 /usr/etc/rpc.rwalld rwalld
mount    dgram  udp wait  root 1 /usr/etc/rpc.mountd mountd
quota    dgram  udp wait  root 1 /usr/etc/rpc.rquotad rquotad
spray    dgram  udp wait  root 1 /usr/etc/rpc.sprayd sprayd
rex      stream tcp wait   root 1 /usr/etc/rpc.rexd   rexd
echo     stream tcp nowait root internal
discard  stream tcp nowait root internal
chargen  stream tcp nowait root internal
daytime  stream tcp nowait root internal
time     stream tcp nowait root internal
```

echo	dgram	udp wait	root	internal
discard	dgram	udp wait	root	internal
chargen	dgram	udp wait	root	internal
daytime	dgram	udp wait	root	internal
time	dgram	udp wait	root	internal

**FILES**

*/etc/inetd.conf* list of internet server processes

**SEE ALSO**

*ftpd(8C)*, *rexecd(8C)*, *rlogind(8C)*, *rshd(8C)*, *telnetd(8C)*, *comsat(8C)*, *crashdump(8)*, *syslog(8)*, *talkd(8)*, *mountd(8C)*, *rexz(8C)*, *rquotad(8C)*, *rstatd(8C)*, *rusersd(8C)*, *rwalld(8C)*, *sprayd(8C)*

**NOTES**

To selectively invoke TCP debugging packet tracing on a per-service basis, consult the manual page of the server program to determine if it accepts an option to turn on debugging.

## NAME

init - process control initialization

## SYNOPSIS

*/etc/init*

## DESCRIPTION

*init* is invoked inside UNIX as the last step in the boot procedure. *init* causes */etc/.initrc* to be executed unconditionally, if it exists and is executable. This facility is intended for operations (such as loading the stripe table with *putst(8)*) which must be completed before */etc/rc* is run. In general, site-specific commands should *not* be placed in */etc/.initrc*. Where possible, general startup tasks should be performed in */etc/rc* and */etc/rc.local* instead.

*init* normally then runs the automatic reboot sequence as described in *reboot(8)*, and if this succeeds, begins multi-user operation. If the reboot fails, it commences single user operation by giving the super-user a shell on the console. It is possible to pass parameters from the boot program to *init* so that single user operation is commenced immediately. When such single user operation is terminated by killing the single-user shell (i.e. by hitting ^D), *init* runs */etc/rc* without the reboot parameter. This command file performs housekeeping operations such as removing temporary files, mounting file systems, and starting daemons.

In multi-user operation, *init*'s role is to create a process for each terminal port on which a user may log in. To begin such operations, it reads the file */etc/ttys* and forks several times to create a process for each terminal specified in the file. Each of these processes opens the appropriate terminal for reading and writing. These channels thus receive file descriptors 0, 1 and 2, the standard input and output and the diagnostic output. Opening the terminal will usually involve a delay, since the *open* is not completed until someone is dialed up and carrier established on the channel. If a terminal exists but an error occurs when trying to open the terminal *init* complains by writing a message using the *syslog(9)* facility. the message is repeated every 10 minutes for each such terminal until the terminal is shut off in */etc/ttys* and *init* notified (by a hangup, as described below), or the terminal becomes accessible (*init* checks again every minute). After an open succeeds, */etc/getty* is called with argument as specified by */etc/ttys* file. *getty* reads the user's name and invokes *login* to log in the user and execute the shell.

Ultimately the shell will terminate because of an end-of-file either typed explicitly or generated as a result of hanging up. The main path of *init*, which has been waiting for such an event, wakes up and removes the appropriate entry from the file *utmp*, which records current users, and makes an entry in */usr/adm/wtmp*, which maintains a history of logins and logouts. The *wtmp* entry is made only if a user logged in successfully on the line. Then the appropriate terminal is reopened and *getty* is reinvoked.

*init* catches the *hangup* signal (signal SIGHUP) and interprets it to mean that the file */etc/ttys* should be read again. The shell process on each line which used to be active in *ttys* but is no longer there is terminated; a new process is created for each added line; lines unchanged in the file are undisturbed. Thus it is possible to drop or add phone lines without rebooting the system by changing the *ttys* file and sending a *hangup* signal to the *init* process: use 'kill -HUP 1.'

*init* will terminate multi-user operations and resume single-user mode if sent a terminate (TERM) signal, i.e. "kill -TERM 1". If there are processes outstanding which are deadlocked (due to hardware or software failure), *init* will not wait for them all to die (which might take forever), but will time out after 30 seconds and print a warning message.

*init* will cease creating new *getty*'s and allow the system to slowly die away, if it is sent a terminal stop (TSTP) signal, i.e. "kill -TSTP 1". A later hangup will resume full multi-user operations, or a terminate will initiate a single user shell. This hook is used by *reboot(8)* and *halt(8)*.

*init*'s role is so critical that if it dies, the system will reboot itself automatically. If, at bootstrap time, the *init* process cannot be located, the system will loop during initialization.

**DIAGNOSTICS**

**init: tty: cannot open.** A terminal which is turned on in the *rc* file cannot be opened, likely because the requisite lines are either not configured into the system or the associated device was not attached during boot-time system configuration.

**init: tty: getty failing, sleeping.** Numerous attempts to start *getty* on the named terminal have failed. Attempts will resume after a 30 second wait. Could be indicative of a malfunctioning port or terminal.

**WARNING: Something is hung (won't die); ps axl advised.** A process is hung and could not be killed when the system was shutting down. This is usually caused by a process which is stuck in a device driver due to a persistent device error condition.

**FILES**

/dev/console, /dev/tty\*, /etc/utmp, /usr/adm/wtmp, /etc/ttys, /etc/rc /etc/.initrc

**SEE ALSO**

login(1), kill(1), sh(1), syslog(3), ttys(5), getty(8), rc(8), reboot(8), halt(8), shutdown(8)

**NAME**

*inline* - inline subroutine call expander for kernel builds

**SYNOPSIS**

*inline* [-cn] [-d] [-v] [ *inputfile* ] [ *outputfile* ]

**DESCRIPTION**

*inline* is used to remove subroutine calls for frequently called kernel functions when building a kernel. *inline* removes subroutine calls by replacing calling sequences with procedure code segments. The input to *inline* is assembly language source; the output of *inline* is the same assembly source with inline replacements made.

*inline* is based on the old *asm.sed* script that replaced instructions of the form *calls \_kernel\_routine* with the (slightly modified) code for *\_kernel\_routine*. *inline* is more intelligent and is not restricted to simple substitutions. *inline* removes the code that precedes and follows a procedure call invocation, including the saving and restoring of register variables. If a procedure is called with arguments, *inline* also alters the calling sequence so that all push operations (*psh.w*, *pshea*) are replaced with move operations. Arguments need not be on the stack; instead, arguments are stored into defined registers.

**EXAMPLE**

For the example, consider the function *\_imin* which finds the minimum of two integer arguments and returns the value in register *s0*. The example shows the definition of the inline procedure for *inline*, the original assembly source, and the output of *inline* after expansions have been made for *\_imin*.

The expansion sequence (taken from langpats.c) is:

```
{ 2, "_imin\n",          /* _imin has two arguments */
"
  pop.w  s1\n\
  pop.w  s2\n\
  lt.w   s1,s2\n\
  brs.f  14$\n\
  mov    s1,s0\n\
  br     18$\n\
14$\n\
  mov    s2,s0\n\
18$\n" },
```

The original assembly code is:

```
; call _imin to find the minimum of a and b
ld.w  -4(fp),s0
psh.w  s0
ld.w  -8(fp),s0
psh.w  s0
mov    sp,ap
pshea  #2
calls  _imin
sub.w  #12,sp
ld.w  12(fp)
```

After *inline* processing, the code becomes:

```

; call _imin to find the minimum of a and b
; optimized
ld.w  -4(fp),s2
; optimized
ld.w  -8(fp),s1
lt.w  s1,s2
brs.f 14$
mov   s1,s0
br    18$
14$:
    mov   s2,s0
18$:
L_end0:

```

In the expansion code, notice that the first argument is specifically “popped” (actually moved via optimizations in the expansion) into *s1*, and the second argument is moved into *s2*. The code for *\_imin* only manipulates *s1* and *s2*. If optimizations cannot be performed, then this code is inserted wholesale and the overhead for the call is still saved.

The temporary labels allow this expansion to be used more than once per source file. The expansion ends with the unique label *L\_end0* so as to undefine the temporary labels.

As *\_imin* was a function, the return value is still returned in *s0*.

#### OPTIONS

The *-cn* flag is one of *-c1* or *-c2* used to specify that the resulting code will be executed on a CONVEX C1 or CONVEX C200 Series System, respectively. *inline* selects between different versions of assembly expansions of the same routine based on this option.

By specifying the *-d* option, the user is given statistics on *inline*'s expansion processing. Upon completion, *inline* gives the number of optimizations attempted, the number of successful replacements done, the number of replacements inhibited, and the number of successful optimizations completed.

The *-v* flag specifies that the assembly language is output from the *vc* compiler.

If no *inputfile* is given, *inline* uses standard input. If no *outputfile* is specified, *inline* sends its output to standard output.

#### NOTES

A few conventions apply when writing the routines for inline insertion.

Temporary registers are limited to *s0*, *s1*, *s2*, and *s3*. The other *s* registers are for register variables; if more temporary registers are needed, specific code must be included that will save register variables in local storage on entry and restore on exit. Address registers may be used at any time.

Functions return values in the *s0* register. If *s0* is used as a temporary register (generally, not necessary), be sure it is set before exiting the procedure code.

The label *L\_end\** is generated by *inline* to undefine temporary labels. Temporary labels are necessary so that exact code sequences may appear several times in one assembly source sequence.

No return instructions (*rtnq*, *rtn*) should appear in the expansion code.

## NAME

installsw – install or create CONVEX software release tapes

## SYNOPSIS

```
/etc/installsw [-r] [-i] [-f in_file] [-d tape_name] [-F]
```

## DESCRIPTION

*installsw* is used to create and install CONVEX software release tapes.

A release tape contains a header file with the CONVEX copyright notice, and other information describing the software release; a script file to perform software installation; and one or more data files to be installed by the installation script. The installation process involves verification of the header file, followed by loading and execution of the installation script file. The script file controls the remainder of the installation process.

To create an install tape, you must specify text for the header and install script, and specify a build script which is executed to control creation of the release tape data. The build script is not written to the tape.

The default creation/installation device for CONVEX software is */dev/rmt12*. This can be changed by the *device* command shown below, or the *-d* option on the command line. It can be changed to the SPU cartridge tape by using the device name *ct*. When running on the SPU, the creation/installation device is the SPU cartridge tape only.

The *-r*, *-i*, and *-f* options are mutually exclusive.

If the *-r* option is used, the *verify* command (see below) is executed noninteractively to verify the tape header. The default device is used unless specified with *-d* option.

If the *-i* option is used, the *install* command (see below) is automatically executed noninteractively. The default device is used unless specified with *-d* option.

If the *-f* option is used, *installsw* takes all further command input from the file *in\_file*. The input must contain valid commands from the list given below. All commands are executed noninteractively. The default device is used unless specified with the *-d* option. Of course, the command file can also respecify the tape device.

*installsw* normally checks the tape device to make sure it really is a tape device before using it. If the *-F* option is used, this check is not made. That is, *installsw* is “forced” to use the tape device regardless of what it is.

If *installsw* is invoked without the *-r*, *-i*, or *-f* commands, it operates in interactive mode. The initial tape device setting is the default device unless changed with the *-d* option on the command line. The user can also specify the tape device at any time with the *device* command.

When in interactive mode, the user is prompted for commands. Command execution may involve considerably more user interaction. This is true even for command files executed from interactive mode with the *auto* command.

The following commands may be input directly to the interactive prompt, or used in command files invoked with the *auto* command, or the *-f* option on the command line.

**d[evice] dev\_name**

This command specifies the device to be used by *install* or *create*. If the device is not set with this command, the default specified in the opening paragraphs applies, unless changed by the *-d* option on the *installsw* command line. If *ct* is specified from the Convex computer, the SPU cartridge tape will be accessed.

In interactive mode, *create* also allows the device name to be changed. *Device* with no arguments will print out the current *dev\_name*.

**h[earer]** *[hd\_file]*

**s[cript]** *[scr\_file]*

**b[uild]** *[bld\_file]*

These commands specify files to be used for header text, install script, and build script when creating a tape. The files are copied to temporary workfiles for later use by the *create* command. The *edit* and *create* commands allow these temporary copies to be edited.

If no filename is given, and *installsw* is in interactive mode, the temporary file is created from keyboard input. Input is terminated with **^D** as the first character in a line. The *edit* command can also be used to create these temporary files.

**e[dit]** **[h]** **[s]** **[b]**

This command allows the user to edit or create the temporary copies of *hd\_file*, *scr\_file*, and *bld\_file*, respectively. More than one option may be specified. If no options are specified, all files will be edited. On the Convex computer system, the editor that is specified in the EDITOR environment string, is used. *vi* is the default editor. On the SPU, the *red* editor is used.

This command can only be used in interactive mode.

**v[erify]** **[h]** **[s]** **[b]** **[t]**

This command displays the contents of the temporary copies of *hd\_file*, *scr\_file*, and *bld\_file*, respectively; or, for the **t** option, the header and script files from the tape at the current *dev\_name*. If no option is given, only the header file from the tape is displayed.

When displaying the header file on tape, the copyright notice, and system generated date/time stamp will be seen prepended to the user's header file.

**c[reate]**

Create an *installsw* tape. If in interactive mode, the user is allowed to edit the work copies of *hd\_file*, *scr\_file*, and *bld\_file*. He is also allowed to change *dev\_name* for Convex computer software. When the user considers all of these data acceptable, **create** writes the temporary copies of the header and script files to the release tape. (The header file is prepended with the standard copyright notice and system date/time stamp). All user interaction is skipped if *installsw* is not in interactive mode. After the header and install script have been written, then the shell script in *bld\_file* is executed. The first argument supplied to the script is the pathname to which the data is to be written. For magnetic tape on the Convex computer, or cartridge tape on the SPU, data files can be written to the device directly or via any utility (such as *tar*). For SPU cartridge tape access from the Convex computer, the *bld\_file* script must create one or more temporary files and use *ctar* to transfer the files to the device at the supplied pathname. *Ctar* allows superuser to write to the SPU cartridge tape.

**i[nstall]**

This command installs a tape. The device used may be set with the *d* command, or the **-d** option on the *installsw* command line. The header file is checked to see if it is a legal *installsw* tape. If so, the header file is displayed on the standard output. The user sees the header file prepended with the standard copyright notice and the date/time stamp when the release tape was created. If in interactive mode, the user is prompted for permission to continue. If the response is yes, the script file is read from the tape and

executed. The first argument *installsw* supplies to the script is the pathname of the installation device. If using standard magnetic tape from the Convex computer, or cartridge tape from the SPU, the data files may be read directly by the script. If using cartridge tape from the Convex computer, the script file should use *ctar* to load the necessary files, and then perform any moving or renaming necessary.

#### a[uto] *in\_file*

This command is the same as specifying the *-f* argument when invoking *installsw*. Further commands are taken from *in\_file*, until an end-of-file is reached. *Auto* commands can be nested. *In\_file* may also be terminated by typing *quit*, or *<.>* on a line by itself.

When *installsw* is in interactive mode, command files are run with full user interaction. (Command files executed by invoking *installsw* with the *-f* option are run noninteractively.)

#### q[uit] and *<period>*

Terminate the current command file. Terminate *installsw* if at the prompt *->*. Command files are automatically terminated by reaching EOF (*quit* or *<period>* in a command file is unnecessary).

In interactive mode *installsw* may also be terminated by responding to the prompt with a *^D <control-D>*.

#### !shell\_cmd

Execute the shell command following the *!*.

? Print a list of the available commands to the standard output.

# The following text is a comment. No processing is performed. This is provided to allow commentary inside command files.

*installsw* sets the file creation mask to 022 (see *umask(2)*). Only the owner will have write access to files created by *installsw*. If this value is unsatisfactory, the umask can be set in the user's *build* and *install* scripts.

### USER BUILD AND INSTALL SCRIPTS

User *bld\_file* and *scr\_file* scripts will be executed by the standard Bourne shell. All facilities for controlling the execution of the shell are described in the shell documents.

*installsw* attempts to gather the exit status from script executions. If error status is returned via "exit" commands, it will be reported, and the *installsw* execution will be terminated. Note that some commands (such as *tar*) that may be used in these scripts do not always report proper status on all exceptions.

*installsw* inherits its environment, *\$PATH*, *\$TERM*, etc., from the user's shell. If the build or install script depends on a special environment, the environment should be set in the build and install scripts.

### DEVICES

When executed on the CONVEX CPU, *installsw* may utilize magnetic tape drives, with */dev/rmt12* as the default drive. It is also possible to access the cartridge tape on the SPU from the Convex computer. This is done by specifying device name *ct*.

When executed on the SPU, *installsw* uses only the SPU cartridge tape drive.

In cartridge tape usage, the header and script files are written from disk files */tmp/install1* and */tmp/install2*, respectively, to file */dev/rct0b*, using *tar* or *ctar*. Therefore, they can be recovered in another *installsw* session by using the same filenames. The install data is written by the *bld\_file* script onto file */dev/rct0e*. When writing from the Convex computer system onto the SPU cartridge tape, the user supplied script must use the *ctar* utility. There may be up to approximately 17 Mbytes of data. Only the superuser may create, read, or install software on a cartridge tape, since only the superuser may execute *ctar*.

#### COMPATIBILITY

This version of *installsw* is fully compatible with the previous version. However, the latest version passes the name of the tape drive as an argument to the user-supplied installation script; the old *installsw* DID NOT. Therefore, installation scripts should be coded to check the argument count, and default to */dev/rmt12* if no tape drive argument is supplied by the version of *installsw* being used. This can be done as shown below:

```
#
# Product foo installation script
#
if $#argv == 0 then
    set drive=/dev/rmt12
else
    set drive=$1
endif
.
    User supplied code
.
tar xvf $drive
```

#### EXAMPLE

As a straightforward method of using *installsw*, the following procedure is suggested. Four files are required:

<i>in_cmds</i>	<i>installsw</i> command input file
Header	Tape header file
Build	Tape build script
Script	Tape install script

The *installsw* command file, *in\_cmds*, should contain the following commands:

```
device    /dev/rmt12
header    Header
script     Script
build     Build
verify    h
create
quit
```

The tape header file, *Header*, can contain any ASCII text and should specify useful information about the tape such as the contents of the tape and the tape release date. For example:

```
Product:    Utilities V2.0
Release date: Jul 1 1985
Directories: /mnt/bin, /mnt/test
```

The tape build script, *Build*, should consist of the commands to build the tape:

```
#!/bin/sh
# Utilities Build Script
#
TAPE=/dev/rmt12      # default tape unit
if [ -n "$1" ]      # is $1 set
then
    TAPE="$1"
fi
tar cvf $TAPE bin test
mt -f $TAPE rew
```

Finally, the tape install script, *Script*, should consist of all commands necessary to install the tape on the target system:

```
#!/bin/sh
# Utilities Installation Script
#
TAPE=/dev/rmt12      # default tape unit
if [ -n "$1" ]      # is $1 set
then
    TAPE="$1"
fi
LOADDIR=/usr
#
cd $LOADDIR
/bin/echo "—" V2.0 Utilities Update
/bin/echo -n "—" V2.0 Utilities installation begun "
/bin/date
#
# load the new release on top of the old one
#
/bin/echo "—" Extracting V2.0 Utilities Update from \'$TAPE\' into \'/bin/pwd\'
/bin/tar xf $TAPE 2>&1 /dev/null
/bin/mt -f $TAPE rew
/bin/echo -n "—" $LOADDIR V2.0 Utilities installation completed "
/bin/date
```

Note that since *installsw* checks the status returned by *Build* and *Script*, it is good practice to include an *exit 0* command at the end of the script. This prevents nonzero exit status from being returned inadvertently as can happen if certain UNIX commands (such as *test*) are the last commands to be executed by the script.

Having established these four files, *installsw* format tapes can be created by executing the command:

```
/etc/installsw -f in_cmds
```

The resultant tape can then be installed on another system via the command:

```
/etc/installsw -i
```

**FILES**

Temporary files are:

*/tmp/Ins\_B\**  
*/tmp/Ins\_H\**  
*/tmp/Ins\_S\**  
*/tmp/Ins\_Y\**  
*/tmp/Ins\_Z\**  
*/tmp/install1*  
*/tmp/install2*

**SEE ALSO**

tar(1), mtio(4), dump(8), restore(8)

**NAME**

**lim** - change share data base information for one or more users

**SYNOPSIS**

**lim** [-p *shares\_file*] *limit*{+|=|=}value[,value...][:...] *name*[*uid-uid*][*uid..* ...

**DESCRIPTION**

*lim* changes records in the share data base file for several users, a range of uids, or a list of users from standard input. If the -p flag is specified *lim* will use *shares\_file* as the share data base file.

The changeable limits are designated by their names as defined in the files *<sys/lnode.h>* and *<shares.h>* as follows:

Name	Type	Description
charge	float	long term account charge
flags	string	lnode flags; the only specifiable value is <i>notshared</i>
lastused	date	account last used
sgroup	string	scheduling group for the account
shares	number	allocated shares
usage	float	usage for scheduling

A limit can be incremented (+), decremented (-), or set (=). For flags, + and = set the flag; - clears the flag.

*value* is interpreted as a number, a date (if the string contains a '/'), or as a string (or as a comma-separated list) depending on the type of limit being changed. Strings can only be set. If *value* is "help", then an explanation of the options available for the given limit will be listed.

Additional limits are specified by a colon separated list.

If any following argument *contains* a '-' then it is interpreted as a range of uids; if the argument ends in trailing dots (eg: 100..) it is interpreted as a range of uids running from the first uid up to the maximum number of registered users. Other arguments may be specified as names or uids.

If no additional arguments are given, *lim* reads names and uids from standard input, one name or uid per line.

A list of valid limits is printed out if *lim* is invoked with invalid arguments (or no arguments).

**EXAMPLES**

To set the share group to "sysstaff" for user "smith":

```
lim sgroup==sysstaff smith
```

To give uids 200 through 400 one hundred shares each:

```
lim shares==100 200-400
```

To turn off the not-shared flag for share group "sysstaff":

```
lim flags-notshared sysstaff
```

**FILES**

/usr/include/sys/share.h	Definition of scheduler parameters.
/usr/include/sys/charges.h	Default scheduler parameters.
/etc/shares	for share details.
/etc/passwd	for user names and IDs.

**SEE ALSO**

pl(1), lnode(5), share(5), shares(5), remshent(8)

**DIAGNOSTICS**

... could not change kernel lnode ...

The limits system call failed for a logged in user, usually because you are attempting to change a scheduling group to one that isn't currently active.

**BUGS**

*lim* does not use *getshput(3)*, so be careful something else is not updating the same entry simultaneously.

**NOTES**

*lim* is part of the Share scheduler. Share is an optional product; please contact your CONVEX sales representative for more information.

## NAME

lockd, rpc.lockd - network lock daemon

## SYNOPSIS

`/etc/rpc.lockd [ -t timeout ] [ -g graceperiod ]`

## DESCRIPTION

*lockd* processes lock requests that are either sent locally by the kernel or remotely by another lock daemon. *lockd* forwards lock requests for remote data to the server site's lock daemon through the *rpc/xdr*(3N) package. *lockd* then requests the status monitor daemon, *statd*(8C), for monitor service. The reply to the lock request will not be sent to the kernel until the status daemon and the server site's lock daemon have replied.

If either the status monitor or server site's lock daemon is unavailable, the reply to a lock request for remote data is delayed until all daemons become available.

When a server recovers, it waits for a grace period for all client site *lockd*'s to submit reclaim requests. Client site *lockd*'s, on the other hand, are notified by the *statd* of the server recovery and promptly resubmit previously granted lock requests. If a *lockd* fails to secure a previously granted lock at the server site, the *lockd* sends SIGLOST to a process.

## OPTIONS

`-t timeout` *lockd* uses *timeout* (seconds) as the interval instead of the default value (15 seconds) to retransmit lock request to the remote server.

`-g graceperiod`

*lockd* uses *graceperiod* (seconds) as the grace period duration instead of the default value (45 seconds).

## SEE ALSO

*fentl*(2), *lockf*(3), *signal*(3c), *statd*(8C)

## NAME

*lpc* – line printer control program

## SYNOPSIS

*/etc/lpc* [ command [ argument ... ] ]

## DESCRIPTION

The *lpc* command is used by the system administrator to control the operation of the line printer system. For each line printer configured in */etc/printcap*, *lpc* may be used to:

- disable or enable a printer,
- disable or enable a printer's spooling queue,
- rearrange the order of jobs in a spooling queue,
- find the status of printers, and their associated spooling queues and printer daemons.
- redirect spool queues locally or globally.

Without any arguments, *lpc* will prompt for commands from the standard input. If arguments are supplied, *lpc* interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected causing *lpc* to read commands from file. Commands may be abbreviated; the following is the list of recognized commands.

? [ command ... ]

help [ command ... ]

Print a short description of each command specified in the argument list, or, if no arguments are given, a list of the recognized commands.

abort { all | printer ... }

Terminate an active spooling daemon on the local host immediately and then disable printing (preventing new daemons from being started by *lpr*) for the specified printers.

clean { all | printer ... }

Remove all files beginning with "cf", "tf", or "df" from the specified printer queue(s) on the local machine.

enable { all | printer ... }

Enable spooling on the local queue for the listed printers. This will allow *lpr* to put new jobs in the spool queue.

exit

quit

Exit from *lpc*.

disable { all | printer ... }

Turn the specified printer queues off. This prevents new printer jobs from being entered into the queue by *lpr*.

down printer

Stop a spooling daemon after the current job completes and disable spooling.

redirect [all] source-printer destination-printer

Redirect the jobs from the source-printer to the destination printer. If *all* is specified, redirect all jobs submitted from anywhere in the network to the source-printer to the destination-printer. Otherwise, redirect only those jobs submitted from the machine on which the redirect is issued. If a local redirection is performed on the source-printer's host machine, all jobs submitted to that printer from the network will be redirected to the destination-printer.

restart { all | printer ... }

Attempt to start a new printer daemon. This is useful when some abnormal condition causes the daemon to die unexpectedly leaving jobs in the queue. *Lpq* will report that there is no daemon present when this condition occurs.

start { all | printer ... }

Enable printing and start a spooling daemon for the listed printers.

status [ all ] [ printer ... ]

Display the status of daemons and queues on the local machine.

stop { all | printer ... }

Stop a spooling daemon after the current job completes and disable printing.

topq printer [ jobnum ... ] [ user ... ]

Place the jobs in the order listed at the top of the printer queue.

undirect [all] printer

Remove the queue redirection from the printer. If *all* is specified, remove the redirection from the printer's host machine. Otherwise, remove the redirection from the local machine.

up printer

Enable the printer and restart the spooling daemon.

#### FILES

/etc/printcap	printer description file
/usr/spool/*	spool directories
/usr/spool/*/lock	lock file for queue control

#### SEE ALSO

lpd(8), lpmv(1), lpr(1), lpq(1), lprm(1), printcap(5)

#### DIAGNOSTICS

?Ambiguous command	abbreviation matches more than one command
?Invalid command	no match was found
?Privileged command	command can be executed by root only

**NAME**

lpd - line printer daemon

**SYNOPSIS**

```
/usr/lib/lpd [-l] [-L logfile] [-f hosts.equivfile] [ port # ]
```

**DESCRIPTION**

*lpd* is the line printer daemon (spool area handler) and is normally invoked at boot time from the *rc(8)* file. It makes a single pass through the *printcap(5)* file to find out about the existing printers and prints any files left after a crash. It then uses the system calls *listen(2)* and *accept(2)* to receive requests to print files in the queue, transfer files to the spooling area, display the queue, or remove jobs from the queue. In each case, it forks a child to handle the request so the parent can continue to listen for more requests. The Internet port number used to rendezvous with other processes is normally obtained with *getservbyname(3)*, but can be changed with the *port#* argument. The *-L* option changes the file used for writing error conditions from the system console to *logfile*. The *-l* flag causes *lpd* to log valid requests received from the network. This can be useful for debugging purposes.

Access control is provided by two means. First, all requests must come from one of the machines listed in the file *hosts.equiv*. Second, if the *rs* capability is specified in the *printcap* entry for the printer being accessed, *lpr* requests will only be honored for those users with accounts on the machine with the printer. The *-f* option changes the default *hosts.equiv* file. The default *hosts.equiv* file is */etc/hosts.equiv*.

The file *lock* in each spool directory is used to prevent multiple daemons from becoming active simultaneously, and to store information about the daemon process for *lpr(1)*, *lpq(1)*, *lpmv(1)*, and *lprm(1)*. After the daemon has successfully set the lock, it scans the directory for files beginning with *cf*. Lines in each *cf* file specify files to be printed or non-printing actions to be performed. Each such line begins with a key character to specify what to do with the remainder of the line.

- J** Job Name. String to be used for the job name on the burst page.
- C** Classification. String to be used for the classification line on the burst page.
- L** Literal. The line contains identification information from the password file and causes the banner page to be printed.
- T** Title. String to be used as the title for *pr(1)*.
- H** Host Name. Name of the machine where *lpr* was invoked.
- P** Person. Login name of the person who invoked *lpr*. This is used to verify ownership by *lprm* and *lpmv*.
- M** Send mail to the specified user when the current print job completes.
- f** Formatted File. Name of a file to print which is already formatted.
- l** Like **f** but passes control characters and does not make page breaks.
- p** Name of a file to print using *pr(1)* a filter.
- t** *Troff* File. The file contains *troff(1)* output (*cat* phototypesetter commands).
- d** DVI File. The file contains *TeX(1)* output (DVI format from Stanford).
- g** Graph File. The file contains data produced by *plot(3X)*.
- c** *Cifplot* File. The file contains data produced by *cifplot*.
- v** The file contains a raster image.
- r** The file contains text data with FORTRAN carriage control characters.
- 1** *Troff* Font R. Name of the font file to use instead of the default.
- 2** *Troff* Font I. Name of the font file to use instead of the default.

- 3**     *Troff* Font B. Name of the font file to use instead of the default.
- 4**     *Troff* Font S. Name of the font file to use instead of the default.
- W**     Width. Changes the page width (in characters) used by *pr*(1) and the text filters.
- I**     Indent. The number of characters to indent the output by (in ASCII).
- U**     Unlink. Name of the file to remove upon completion of printing.
- N**     File name. The name of the file that is being printed, or a blank for the standard input when *lpr* is invoked in a pipeline.
- A**     Accounting information: user ID, group ID, activity ID, and printer name. This information is present in the control file if the *pu* capability is set in the selected printer's printcap entry. *lpd* propagates this information to whatever print filter is used to process the job. See *lpd-acct*(5) for more information.

If a file can not be opened, a message will be placed in the log file (normally the console). *lpd* will try up to 20 times to reopen a file it expects to be there, after which it will skip the file to be printed.

*lpd* uses *flock*(2) to provide exclusive access to the lock file and to prevent multiple daemons from becoming active simultaneously. If the daemon should be killed or die unexpectedly, the lock file need not be removed. The lock file is kept in a readable ASCII form and contains two lines. The first is the process ID of the daemon; the second is the control file name of the current job being printed. The second line is updated to reflect the current status of *lpd* for the programs *lpq*(1), *lprm*(1), and *lpmv*(1).

#### FILES

<i>/etc/printcap</i>	printer description file
<i>/usr/spool/*</i>	spool directories
<i>/dev/lp*</i>	line printer devices
<i>/dev/printer</i>	socket for local requests
<i>/etc/hosts.equiv</i>	lists machine names allowed printer access

#### SEE ALSO

*lpmv*(1), *lpq*(1), *lpr*(1), *lprm*(1), *hosts.equiv*(5), *lpd-acct*(5), *printcap*(5) *lpc*(8), *lprewind*(8), *pac*(8), "4.2BSD Line Printer Spooler Manual" in the *CONVEX UNIX Tutorial Papers*, "Accounting" chapter in the *CONVEX System Manager's Guide*.

## NAME

*lpf* – general line printer filter

## SYNOPSIS

**lpf** [ **-c** ] [ **-h** *host* ] [ **-i** *indent* ] [ **-l** *length* ] [ **-n** *name* ] [ **-u** *userinfo* ] [ **-w** *width* ] [ *acctfile* ]

## DESCRIPTION

*lpf* is a general-purpose filter for use with the line printer spooler system. It is not meant to be invoked directly by a user (the filter is normally specified in the *printcap*(5) file with the “if” option, and invoked by *lpd*(8) when necessary).

The following options are understood by *lpf*:

- c** Print control characters, instead of processing them out of the text.
- h** *host* Specify the name of the host from which the file came.
- i** *indent* Specify the indent level (left margin).
- l** *length* Specify the length of the page to be printed. The default is 66 lines per page.
- n** *name* Specify the login name of the user who printed the file.
- u** *userinfo* Specify user information, which is included as part of the log entry for this job when the job is logged in the accounting file. This information consists of a user ID, a group ID, an activity ID, and a printer name, and is passed as a single command-line argument with each field separated by a space. This information is passed to *lpf* by *lpd*(8) if the *pu* capability (propagate user information) is set in */etc/printcap*(5).
- w** *width* Specify the width of the page to be printed. The default is 132 columns per page.
- acctfile* Specify the name of a file to receive printer accounting information. This information may be subsequently read and processed by *pac*(8). *acctfile* must be the last argument on the command line.

Besides performing printer accounting, *lpf* processes and correctly prints overstruck lines generated by *nroff*(1), much in the same way that *ul*(1) does.

Some sites may wish to write their own print filters, based on their accounting needs. As a general guide for writing site specific filters, the */usr/lib/filter* directory contains a sample filter program written in C, a compiled object form of the *log* routine, a sample makefile (see *make*(1)), and a README file which describes each of these files in more detail. These files are intended to be used only as a template.

## SEE ALSO

*lpr*(1), *nroff*(1), *printcap*(5)  
 “4.2BSD Line Printer Spooler Manual” in the *CONVEX UNIX Tutorial Papers*

## BUGS

If *lpf* is used, *lprm*(1) cannot be used to kill print jobs.

## NAME

lprewind, lprew-daemon, lpf-rew - line printer rewind commands

## SYNOPSIS

```
/usr/lib/lprewind numpages printer
/usr/lib/lprew-daemon printer
/usr/lib/lpf-rew [-c] [-d printer] [-h host] [-llength] [-n name] [-wwidth] [acctfile]
```

## DESCRIPTION

*Lprewind*, *lprew-daemon*, and *lpf-rew* are utilities used to give a *dumb* printer a buffer of 10 printed pages. *Lprew-daemon* is a daemon process whose task is to remember the last 10 pages sent to the line printer and write data to the printing device *printer*. *Lprew-daemon* should be invoked in */etc/rc.local*.

*Lprewind* is a program that interrupts printing on the device specified in *printer*, tells *lprew-daemon* to re-print the previous *numpages* pages (where *numpages* is a number between 1 and 10), and resumes normal printer operation.

*Lpf-rew* is a general-purpose filter for use with the line printer spooler rewind system. It is used in conjunction with *lprew-daemon* and *lprewind*. Contrary to normal line printer filter actions, it sends data to a socket named *printer-sock* rather than directly to the printing device *printer*. It is not meant to be invoked directly by a user (the filter is normally specified in the *printcap*(5) file with the *if* option, and invoked by *lpd*(8) when necessary). An example of a *printcap*(5) entry using *lpf-rew* is:

```
lp|local line printer:\
:af=/usr/adm/lpd-acct:if=/usr/lib/lpf_rew.sh:mx=0:\
:lp=/dev/null:sd=/usr/spool/lpd:lf=/usr/adm/lpd-errs:
```

Notice that the line printer device in *printcap* is specified as */dev/null*. This is done because data are sent to the socket *printer-sock* not to the actual printing device. If the actual printing device had been specified, *lpf-rew* would fail; two processes cannot open the printing device simultaneously.

*Lpf\_rew.sh* is a shell script that invokes *lpf-rew* with an additional argument not specified by *lpd*(8). The text of *lpf\_rew.sh* is:

```
#!/bin/sh
/usr/lib/lpf-rew -d /dev/lp $*
```

The following options are understood by *lpf-rew*:

- c Print control characters, instead of processing them out of the text.
- d *printer* Specify the name of the printing device.
- h *host* Specify the name of the host from which the file came.
- l*length* Specify the length of the page to be printed. The default is 66 lines per page.
- n *name* Specify the login name of the user who printed the file.
- w*width* Specify the width of the page to be printed. The default is 132 columns per page.
- acctfile* Specify the name of a file to receive printer accounting information. This information may be subsequently read and processed by *pac*(8). *Acctfile* must be the last argument on the command line.

Besides performing printer accounting, *lpf-rew* processes and correctly prints overstruck lines generated by *nroff*(1), much in the same way that *ul*(1) does.

**FILES**

/dev/lp*	line printer devices
/dev/*-sock	line printer rewind socket
/dev/*-rew	line printer rewind socket
/etc/printcap	printer description file
/etc/rc.local	local additions to reboot file
/usr/spool/lpd/lprewind.lock	lprew-daemon lock file
/usr/spool/lpd/logfile	lprew-daemon logfile

**SEE ALSO**

*lpr*(1), *nroff*(1), *ul*(1), *printcap*(5), *lpd*(8), *rc*(8)

**BUGS**

*Lpf-rew* prints the burst page for each print job but does not know the jobname for the job.

The burst page is included as a page in the memory of *lprew-daemon*.

The interface between *lprew-daemon*, *lprewind*, *lpf-rew* utilities and *lpd* is cumbersome and should be integrated into the functionality of *lpd*.

**NAME**

madev, MAKEDEV – make system special files

**SYNOPSIS**

`/dev/MAKEDEV [NAME...]`

**DESCRIPTION**

*MAKEDEV* is a shell script normally used to install special files. It resides in the */dev* directory, as this is the normal location of special files. Since all devices are created using *mknod*(8), this shell script is useful only to the super-user.

**NAME** is the name of the devices to be created. Valid arguments are -

<i>Name</i>	<i>Description</i>
std	Standard devices
local	Configuration specific devices
ta*	Multibus and VMEbus tapes (MTC-001, MTC-201).
tc*	VME 3480 tapes (MTC-202)
da*	Multibus disks (DKC-001, DKC-002)
dd*	VMEbus disks (DKC-203 DKC-204)
du*	IDC disks (DKC-IP2)
ca*	Multibus and VMEbus terminal multiplexors (ACM-001, ACM-201)
pty*	set of 16 master and slave pseudo terminals
pa*	Multibus and VMEbus printers (PRC-001, PTR-CEN, PTR-DAT)
pb*	Versatec Plotter (PRC-002)
dm*	Parallel Interface to Raster Tech Model One/80 (GPI-001)

**DIAGNOSTICS**

Either self-explanatory, or generated by one of the programs called from the script. Use “sh -x MAKEDEV” in case of trouble.

**SEE ALSO**

intro(4), mknod(8)

**NAME**

makekey - generate encryption key

**SYNOPSIS**

**/usr/lib/makekey**

**DESCRIPTION**

*Makekey* improves the usefulness of encryption schemes depending on a key by increasing the amount of time required to search the key space. It reads 10 bytes from its standard input, and writes 13 bytes on its standard output. The output depends on the input in a way intended to be difficult to compute (that is, to require a substantial fraction of a second).

The first eight input bytes (the *input key*) can be arbitrary ASCII characters. The last two (the *salt*) are best chosen from the set of digits, upper- and lower-case letters, and '.' and '/'. The salt characters are repeated as the first two characters of the output. The remaining 11 output characters are chosen from the same set as the salt and constitute the *output key*.

The transformation performed is essentially the following: the salt is used to select one of 4096 cryptographic machines all based on the National Bureau of Standards DES algorithm, but modified in 4096 different ways. Using the input key as key, a constant string is fed into the machine and recirculated a number of times. The 64 bits that come out are distributed into the 66 useful key bits in the result.

*Makekey* is intended for programs that perform encryption (for instance, *ed* and *crypt(1)*). Usually *makekey*'s input and output will be pipes.

**SEE ALSO**

crypt(1), ed(1)

## NAME

makewhatis - rebuild the whatis database

## SYNOPSIS

```
/usr/lib/makewhatis [-v] [-n] [-y] [[-M] manpath ]
```

## DESCRIPTION

The *makewhatis* program rebuilds the text and *dbm*(3X) forms of the *whatis* database from the on-line manual pages for a given *manpath*, or from */usr/man* if none is supplied. These files are used by the *man*(1), *whatis*(1), and *apropos*(1) programs to locate man pages and to print out the one-line descriptions given by *whatis* and *apropos*. The *makewhatis* program should be run whenever new man pages are added so these programs can find them.

Each component of the colon-delimited *manpath* is interpreted to be the root of a new tree containing directories of the form *man\**, within which are unformatted man pages of the form *\*.\**. A separate *whatis* file and corresponding *dbm* files, *whatis.pag* and *whatis.dir*, will be placed at the root of that man tree. Files or directories ending in tilde (~) or in *.bak* or *.old* (in either case) will be skipped, as will a section named *man0*, should it exist.

In general, files in section *manX* should end in *.X\**, where *X* is a section of the manual like *1* or *3x11*. This means it's ok to put things ending in *1c* in *man1*, but not things ending in *3s*. Any man page whose name doesn't match these criteria will generate a warning, but will be processed anyway. The exception to this is the *mano* subdirectory, which has traditionally been the receptacle of all obsolete man pages irrespective of their file extension. A better way to do that would be to have an entire man tree dedicated to this, as in */usr/old/man/*.

Multiple entries occurring in the NAME section or as links (hard, soft, or via *.so* inclusion) will all be stored under the same man page name. This method can save significant amounts of disk space because it guarantees that only one cat page need be generated, regardless of how many ways you can get at the corresponding man page. Maintaining aliases as links or via a one-line file that *.so*'s the real man page (once the only way to do this) is still supported although no longer required; mere inclusion in the NAME section is sufficient.

Embedded point and font changes will be removed from the output, and *troff* string macros for hyphens and underscores will be translated into their corresponding ASCII representations. In general, all that can be done will be done to produce nicely readable output for *whatis*(1).

The *-v* flag will generate verbose output, reporting such things as each new man tree examined, each subdirectory, each file that is opened, and each entry in the NAME section that is found and stored. Simple tracing information is printed to the standard output, while to standard error are directed more serious warnings.

The *-n* option can be used to check whether the database needs rebuilding. It will report on the first file in each man path that is out of date. Note that this is the only automated way to determine this; neither *makewhatis* nor *man* will realize that the database is out of date, so you should be careful to rebuild it whenever new man pages are added.

The *-y* option is similar, but after finding something out of date, *makewhatis* will automatically rebuild its database.

Support for compressed man pages is provided in the following way: if the name of the subdirectory itself ends in *.Z*, as in *man8.Z*, then all its files are assumed to have been compressed with *compress*(1L). Alternatively, individual files ending in *.Z* are also considered to be compressed. Compressed files are processed with *zcat*(1L).

## EXAMPLES

```
% makewhatis -y                # build /usr/man database only if out of date
% makewhatis /usr/local/man    # make local man page database
% makewhatis -v ~/man          # make personal man page database verbosely
% makewhatis -n $MANPATH       # tell if any dbase in $MANPATH out of date
```

## FILES

<code>/usr/man/whatis</code>	default <i>whatis</i> database, text version
<code>/usr/man/whatis.pag</code>	<i>dbm</i> index file for default <i>whatis</i> database
<code>/usr/man/whatis.dir</code>	<i>dbm</i> data file for default <i>whatis</i> database
<code>/usr/man/man*/*.*</code>	default (unformatted) man pages
<code>/usr/man/cat*/*.*</code>	formatted man pages

## SEE ALSO

man(1), *whatis*(1), *apropos*(1), *perl*(1), *compress*(1L), *dbm*(3X), man(7), *catman*(8)

## DIAGNOSTICS

Numerous diagnostics may be issued, especially if the `-v` flag has been given. Here are messages that may require further attention. The `%s` and `%d` fields in the descriptions indicate strings and integers respectively that will be filled in appropriately at run time, while `%m` indicates a standard system error message as described in *intro*(2). See the source for further details.

## Skipping non-man file: %s

A file was found in a man directory that has no dot in its name.

## %s has a funny extension to be in %s

An apparent man file was found whose extension doesn't match the name of the man file in which it was found. This restriction does not apply to **mano** subdirectories.

## can't stat %s: %m

The *stat*(2) call returned an error. This can be caused by a symbolic link pointing to a non-existent file.

## can't open %s: %m

The *open*(2) call returned an error, which is listed.

## can't chdir back to %s: %m

After changing directory to one of the man subdirectories, *makewhatis* couldn't return to the initial root man tree. This is a fatal error.

## makewhatis: %s: found %d entries in %d files

For the given root man directory, this many separate entry points were found for this many different files.

## %s .so references non-existent %s

A file contains a `.so` reference to a missing man page.

## %s's .TH thinks it's " " "ConvexOS V9.0 Programmer's Reference" " " "ConvexOS Programmer's Reference"

The man page's internal `TH` section has a different idea of where it lives than the actual file name.

## trimmed troff string macro in NAME section of %s

An irresolvable *troff* string macro was found within the `NAME` section of the man page. This may cause peculiar *whatis* output.

## %s: no separated dash in %s

The `NAME` section contained no dash in it to separate the list of man entries from their descriptions. This will also cause odd *whatis* output.

## %s: truncating cmdlist from %d to %d bytes for DBM's sake

The NAME section contained too many characters. Due to built-in limitations of *dbm(3X)* data entries, this entry was truncated. It will appear in *whatis* output with a trailing . . . at the point of truncation.

**%s: forgot my own name!**

A man page is stored in a file name that doesn't correspond to any of the entries in its NAME section.

**%s: no NAME lines, so has no *whatis* description!**

No parsable entries in the NAME section were discovered.

**%s was a .so alias for %s, but %s's NAME section doesn't know it!**

A .so reference was found to point to a man page whose own NAME section didn't contain the name of the .so file. This may be due to old man page aliases that were never de-installed when the base man page changed.

**can't store %s: would break DBM**

There were too many entries for a topic for it to be stored without exceeding inherent *dbm(3X)* data size limitations.

#### NOTES

This version of *makewhatis* is written entirely in the *perl* programming language; it requires that *perl* be installed on the system to run.

#### RESTRICTIONS

The NAME section should be a comma-separated list of aliases for this man page, followed by white space, a dash, more white space, and then the description. Pages not conforming to this rule will still be found, but a diagnostic will be printed and their descriptions will be left blank.

Man pages should be formatted using the *man(7)* macro set. The only exceptions to this are the man pages from the *Rand MH Message Handling System*, whose own internal macro set is also recognized, and Larry Wall's .Sh section header macro.

*makewhatis* takes much longer to run if man pages are stored in compressed form.

#### BUGS

Not all systems have *compress* installed on them.

#### AUTHOR

Tom Christiansen <*tchrist@convex.com*>

#### COPYRIGHT

Copyright 1990 CONVEX Computer Corporation. All rights reserved.

## NAME

mkfs - construct a file system

## SYNOPSIS

```
/etc/mkfs [-E pattern] special size [nsect] [ntrack] [blksize] [fragsize] [maxcontig]
[rotdelay] [ncpg] [minfree] [rps] [nbpi] [sectspare]
```

## DESCRIPTION

**NOTE:** file systems are normally created with the *newfs(8)* and *newst(8)* commands.

*Mkfs* constructs a file system by writing on the special file *special*. *Size* specifies the number of physical sectors in the file system. *Mkfs* builds a file system with a root directory and a *lost+found* directory (see *fsck(8)*). The number of i-nodes is calculated as a function of the file system size. No boot program is initialized by *mkfs* (see *newfs(8)*).

The optional arguments allow fine tune control over the parameters of the file system. Since the file system parameters are non-trivially inter-related, a complete yet simple description of acceptable values for them is not possible. *Mkfs* does the necessary checking for compatibility, and prints specific error messages when incompatible parameters are given.

The **-E pattern** flag specifies that the entire volume should be overwritten with the *pattern* before the file system is initialized. *Pattern* is a 32 bit hex number.

Note that each of the other arguments may optionally be specified, but, if you specify one option, you must also specify all the options to the left of that one.

*Nsect* specifies the number of physical sectors per track on the disk.

*Ntrack* specifies the number of tracks per cylinder on the disk.

*Blksize* gives the primary block size for files on the file system. It must be a power of two within the range 4096 and 65536 (4Kbytes and 64Kbytes).

*Fragsize* gives the fragment size for files on the file system. The *fragsize* represents the smallest amount of disk space that will be allocated to a file. It must be a power of two within the range 1/8th *blksize* and *blksize*.

*Maxcontig* sets the default for the maximum number of blocks that may be allocated sequentially. Since UNIX drivers are not capable of scheduling multi-block transfers, this defaults to 1.

*Rotdelay* gives the minimum number of milliseconds to initiate another disk transfer on the same cylinder. It is used in determining the rotationally optimal layout for disk blocks within a file; the default is 4.

*Ncpg* specifies the number of disk cylinders per cylinder group. This number must be in the range 1 to 32.

*Minfree* specifies the minimum percentage of free disk space allowed. Once the file system capacity reaches this threshold, only the superuser is allowed to allocate disk blocks. The default value is 10%.

If a disk does not revolve at 60 revolutions per second, the *rps* parameter may be specified. Users with special demands for their file systems are referred to the paper cited below for a discussion of the tradeoffs in using different configurations.

*Nbpi* specifies the density of i-nodes in the file system. The default is to create an i-node for each 2048 bytes of data space. If fewer i-nodes are desired, a larger number should be used; to create more i-nodes a smaller number should be specified.

*sectspare* specifies the number of spare physical sectors per cylinder. This is required by the IDC.

## SEE ALSO

fs(5), dir(5), fsck(8), newfs(8), newst(8)

McKusick, Joy, Leffler; "A Fast File System for Unix", Computer Systems Research Group, Dept of EECS, Berkeley, CA 94720; TR #7, September 1982.

**BUGS**

There should be some way to specify bad blocks.

File systems are currently limited to 2 gigabytes in size.

**NOTES**

Historically, disk sectors have always been 512 bytes in size. With the advent of the IDC, however, sector sizes are now variable. Therefore, *size*, *nsect*, and *sectspare* should be expressed in terms of the physical sector size of the device, not in terms of 512 byte units.

**NAME**

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

**SYNOPSIS**

**/etc/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(8)*. This command should not normally be needed since *mkfs(8)* automatically creates the *lost+found* directory when a new file system is created.

**SEE ALSO**

fsck(8), mkfs(8)

## NAME

`mknf`, `rmnf` – create and delete notesfiles

## SYNOPSIS

`mknf` [ `-aon` ] `topic` [ ... ]

`rmnf` [ `-f` ] `topic` [ ... ]

## DESCRIPTION

`mknf` and `rmnf` create and delete notesfiles respectively. The same parameters apply for each: the “topic” is the name by which the notesfile is known.

As `mknf` processes its arguments, creating new notesfiles, the name of each new notesfile is echoed to the terminal. The new notesfiles are closed and the *notesfile owner* is made the sole director. He customarily turns control over to the user requesting the notesfile by making that person a director.

The `-aon` options apply to `mknf` only. They signify that the notesfiles created are to permit anonymous notes, be open, and be networked respectively.

If the file “/usr/spool/notes/.utilities/access-template” is present, it contains a list of access-rights which are added to the created notesfile. The file contains lines of access-rights similar to those used in the `nfacess(8)` command. In environments with dynamic creation of notesfiles, such as on USENET nodes, one or more users can automatically be made directors of all new notesfiles. If appropriate, they can be removed from the access list of newly created notesfiles as needed.

`rmnf` asks for verification of each notesfile before deleting it. The notesfile is deleted if the response line begins with a “y”. If invoked with the `-f` option, `rmnf` does not ask for verification before deleting the notesfiles.

Only the *notesfile owner* is allowed to run `mknf` and `rmnf`.

## BUGS

`rmnf` doesn’t understand about absolute pathnames for notesfiles. It refuses to remove notesfiles specified by absolute pathnames.

Any user should be allowed to create private notesfiles in directories where he has permission.

## FILES

/usr/spool/notes/.utilities	where these programs live.
/usr/spool/notes/.utilities/access-template	Default access-list.
/usr/spool/notes	Default notesfile directory.

## SEE ALSO

`nfacess(8)`, `nfxmit(8)`, `notes(1)`, `notesadm(8)`  
 “Notesfile Reference Manual” in the *ConvexOS Tutorial Papers*

**NAME**

*mknod* - build special file

**SYNOPSIS**

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

**DESCRIPTION**

*mknod* makes a special file. The first argument is the *name* of the entry. In the first form, the second argument is **b** if the special file is block-type (disks, tape) or **c** if it is character-type (other devices, also called *raw* devices.) The last two arguments are numbers specifying the *major* device type and the *minor* device (e.g. unit, drive, or line number). Only the super-user is permitted to invoke this form of the *mknod* command.

In the second form, *mknod* makes a named pipe (fifo).

The first form of *mknod* is only for use by a system administrator in order to configure a custom device driver into the kernel. Normally you should use */dev/MAKEDEV* instead when making special files.

The assignment of major device numbers is specific to each system release. For the 8.0 release of ConvexOS, the assignments are:

**Block Devices**

<i>Device Number</i>	<i>Device Type</i>	<i>Filename</i>
0 - 2	reserved for future use by CONVEX	
3	interface to swap space	<i>/dev/drum</i>
4	magnetic tape	<i>/dev/mtx</i>
5	Multibus disk partitions, units 0-31	<i>/dev/dax</i>
6	reserved for future use by CONVEX	
7	striped disk partitions	<i>/dev/stx</i>
8 - 15	reserved for future use by CONVEX	
16 - 27	reserved for user written device drivers	
28	VME disk partitions, units 0-31	<i>/dev/ddx</i>
29	VME disk partitions, units 32-63	<i>/dev/ddx</i>
30	VME disk partitions, units 64-95	<i>/dev/ddx</i>
31	VME disk partitions, units 96-127	<i>/dev/ddx</i>
32 - 47	reserved for user written device drivers	
48	Multibus disk partitions, units 32-63	<i>/dev/dax</i>
49	Multibus disk partitions, units 64-95	<i>/dev/dax</i>
50	Multibus disk partitions, units 96-127	<i>/dev/dax</i>
51 - 60	reserved for future use by CONVEX	
61	VME Ultranet controller interface	<i>/dev/uvx</i>
62 - 63	reserved for future use by CONVEX	
64	IDC disk partitions, units 0-511	<i>/dev/dux</i>
65 - 127	reserved for future use by CONVEX	
128- 254	reserved for Special Systems products	

**Character Devices**

<i>Device Number</i>	<i>Device Type</i>	<i>Filename</i>
0	system console	/dev/console
1	Channel Control Unit minor device is CCU number	/dev/ccuz
2	logical memory devices minor device 0 : memory minor device 1 : kernel memory minor device 2 : the bit bucket	/dev/mem /dev/kmem /dev/null
3	terminal driver, units 0-255	/dev/ttyz
4	raw tape /dev/rmtz	
5	Multibus raw disk partitions, units 0-31	/dev/rdaz
6	Line printer units	/dev/lpz
7	subdevice of disk driver	/dev/drum
8	indirect tty driver	/dev/tty
9	pseudo terminal lines	/dev/typz
10	pseudo terminal lines	/dev/ptypz
11	interface to ud driver	/dev/udz
12	raw striped disk partitions	/dev/stz
13	DR-11W emulator (unsupported)	/dev/dmz
14	Versatec(tm) Plotter interface	/dev/pbz
15	raw Multibus HYPERchannel driver	/dev/hyz
16	HSP/HIA/ECHO test driver minor device 0-15 is logical channel number	/dev/hspz
17	terminal driver, units 256-511	/dev/ttyz
18	Reserved for X.25 devices	
19	raw Ultra network driver, units 0-255	/dev/unetz
20	net char device - kernel rpc interface	
21	net char device - kernel rpc interface	
22 - 27	reserved for future use by CONVEX	
28	VME raw disk partitions, units 0-31	/dev/rdaz
29	VME raw disk partitions, units 32-63	/dev/rdaz
30	VME raw disk partitions, units 64-95	/dev/rdaz
31	VME raw disk partitions, units 96-127	/dev/rdaz
32 - 47	reserved for user written device drivers	
48	Multibus raw disk partitions, units 32-63	/dev/rdaz
49	Multibus raw disk partitions, units 64-95	/dev/rdaz
50	Multibus raw disk partitions, units 96-127	/dev/rdaz
51	COVUEnet controller download interface	/dev/xmem
52	COVUEnet network manager interface	/dev/netman
53	COVUEnet logical link interface	/dev/ll
54 - 60	reserved for future use by COVUEnet	
61	raw VME Ultramet controller interface	/dev/rvuz
62 - 63	reserved for future use by CONVEX	
64	IDC raw disk partitions, units 0-511	/dev/duz
65 - 127	reserved for future use by CONVEX	
128- 254	reserved for Special Systems products	
255	reserved for NFS	

**DIAGNOSTICS**

*mknod* exits with a status of 0 if successful, a status of 1 for usage errors and a status of 2 for other errors.

**SEE ALSO**

*mknod*(2) *makedev*(8)

## NAME

`mkpasswd` - generate hashed password table(s)

## SYNOPSIS

```
/etc/mkpasswd [ -v ] [ -p passwdfile ] [ -r restfile ]
```

## DESCRIPTION

*Mkpasswd* is used to generate the hashed password database used by the library routines `getpwnam()` and `getpwuid()`, and/or the password restrictions database used by the library routines `getpwrestnam()` and `getpwrestuid()`.

If the `-v` option is supplied, each entry will be listed as it is added. If the `-p` option is supplied, the file *passwdfile* will be used instead of the default `/etc/passwd` to create the passwords database. The file *passwdfile* must be in the format of `/etc/passwd` (see `passwd(5)`). If the `-r` option is supplied, the file *restfile* will be used instead of the default `/etc/pwrestrict` to create the restrictions database. The file *restfile* must be in the format of `/etc/pwrestrict` (see `pwrestrict(5)`).

If no options are given *mkpasswd* will use `/etc/passwd` to generate the passwords database. *Mkpasswd* uses a lock file `/tmp/mkpasswd.lock` to ensure that only one process at a time is running *mkpasswd*. *Mkpasswd* will generate database files named *passwdfile.pag* and *passwdfile.dir* and/or *restfile.pag* and *restfile.dir*. *Mkpasswd* will exit with a non-zero exit code if any errors are detected.

## FILES

*passwdfile.pag* - database filenames  
*passwdfile.dir*  
*restfile.pag*  
*restfile.dir*  
`/tmp/mkpasswd.lock`- lock file

## SEE ALSO

`getpwent(3)`, `getpwrestent(3)`, `vipw(8)`, `passwd(5)`, `pwrestrict(5)`

## NAME

mount, umount – mount and dismount filesystems

## SYNOPSIS

```
/etc/mount [ -p ]
/etc/mount -a[fnv] [ -t type ]
/etc/mount [ -fnrv ] [ -t type ] [ -o options ] fsname dir
/etc/mount [ -vfn ] [ -o options ] fsname | dir

/etc/umount [ -t type ] [ -h host ]
/etc/umount -a[v]
/etc/umount [ -v ] fsname | dir ...
```

## DESCRIPTION

*mount* announces to the system that a filesystem *fsname* is to be attached to the file tree at the directory *dir*. The directory *dir* must already exist. It becomes the name of the newly mounted root. The contents of *dir* are hidden until the filesystem is unmounted. If *fsname* is of the form “host:path”, the filesystem type is assumed to be *nfs*.

*umount* announces to the system that the filesystem *fsname* previously mounted on directory *dir* should be removed. Either the filesystem name or the mounted-on directory may be used.

*mount* and *umount* maintain a table of mounted filesystems in */etc/mtab*, described in *mtab(5)*. If invoked without an argument, *mount* displays the table. Thus the order in which the file systems are listed will be in the order of */etc/mtab*, not */etc/fstab*. If invoked with only one of *fsname* or *dir*, *mount* searches the file */etc/fstab* (see *fstab(5)*) for an entry whose *dir* or *fsname* field matches the given argument. For example, if this line is in */etc/fstab*:

```
/dev/da0e /usr 4.2 rw 1 2
```

then the commands **mount /usr** and **mount /dev/da0e** are shorthand for **mount /dev/da0e /usr**

## MOUNT OPTIONS

- p Print the list of mounted filesystems in a format suitable for use in */etc/fstab*.
- a Attempt to mount all the filesystems described in */etc/fstab*. (In this case, *fsname* and *dir* are taken from */etc/fstab*.) If a type is specified all of the filesystems in */etc/fstab* with that type is mounted. Filesystems are not necessarily mounted in the order listed in */etc/fstab*.
- f Fake a new */etc/mtab* entry, but do not actually mount any filesystems.
- n Mount the filesystem without making an entry in */etc/mtab*.
- v Verbose — *mount* displays a message indicating the filesystem being mounted.
- t The next argument is the filesystem type. The accepted types are: **4.2**, and **nfs**; see *fstab(5)* for a description of these filesystem types.
- r Mount the specified filesystem read-only, even if the entry in */etc/fstab* specifies that it is to be mounted read-write. This is a shorthand for:

```
mount -o ro fsname dir
```

Physically write-protected and magnetic tape filesystems must be mounted read-only, or errors occur when access times are updated, whether or not any explicit write is attempted.

- o Specify *options*, a list of comma separated words from the list below. Some options are valid for all filesystem types, while others apply to a specific type only.

*options* valid on *all* (both **4.2** and **nfs**) file systems (the default is **rw,suid,noquota**):

<b>rw</b>	read/write.
<b>ro</b>	read-only.
<b>suid</b>	set-uid execution allowed.
<b>nosuid</b>	set-uid execution not allowed.
<b>grpuid</b>	Create files with BSD semantics for the propagation of the group ID. Under this option, files inherit the GID of the directory in which they are created, regardless of the directory's set-GID bit.
<b>quota</b>	usage limits enforced.
<b>noquota</b>	usage limits not enforced.
<b>hide</b>	ignore this entry during a <b>mount -a</b> command to allow you to define <i>fstab</i> entries for commonly used filesystems you don't want to automatically mount.
<b>noauto</b>	a synonym for <b>hide</b> .
<b>remount</b>	If the file system is currently mounted, and if the entry in <i>/etc/fstab</i> specifies that it is to be mounted read-write or <b>rw</b> was specified along with <b>remount</b> , remount the file system making it read-write. If the entry in <i>/etc/fstab</i> specifies that it is to be mounted read-only and <b>rw</b> was not specified, the file system is not remounted. If the file system is not currently mounted, an error results.

*options* specific to **nfs** (NFS) file systems (the defaults are:

**fg, retry=1000, timeo=7, retrans=3, port=NFS\_PORT, hard, intr**

with defaults for *rsize* and *wsize* set by the kernel):

<b>bg</b>	if the first mount attempt fails, retry in the background.
<b>fg</b>	retry in foreground.
<b>retry=<i>n</i></b>	set number times to retry mount to <i>n</i> .
<b>rsize=<i>n</i></b>	set read buffer size to <i>n</i> bytes.
<b>wsize=<i>n</i></b>	set write buffer size to <i>n</i> bytes.
<b>timeo=<i>n</i></b>	set NFS timeout to <i>n</i> tenths of a second.
<b>retrans=<i>n</i></b>	set number of NFS retransmissions to <i>n</i> .
<b>port=<i>n</i></b>	set server IP port number to <i>n</i> .
<b>soft</b>	return error if server doesn't respond.
<b>hard</b>	retry request until server responds.
<b>intr</b>	allow keyboard interrupts on hard mounts.
<b>nointr</b>	don't allow keyboard interrupts on hard mounts.
<b>secure</b>	Use a more secure protocol for NFS transactions.
<b>acregmin=<i>n</i></b>	Hold cached attributes for at least <i>n</i> seconds after file modification.
<b>acregmax=<i>n</i></b>	Hold cached attributes for no more than <i>n</i> seconds after file modification.
<b>acdirmin=<i>n</i></b>	Hold cached attributes for at least <i>n</i> seconds after directory update.
<b>acdirmax=<i>n</i></b>	Hold cached attributes for no more than <i>n</i> seconds after directory update.

**actimeo=*n*** Set *min* and *max* times for regular files and directories to *n* seconds.

## NFS FILESYSTEMS

### Background vs. Foreground

The **bg** option causes *mount* to run in the background if the server's *mountd*(8C) does not respond. *mount* attempts each request **retry=*n*** times before giving up. Once the filesystem is mounted, each NFS request made in the kernel waits **timeo=*n*** tenths of a second for a response. If no response arrives, the time-out is multiplied by **2** and the request is retransmitted. When **retrans=*n*** retransmissions have been sent with no reply a **soft** mounted filesystem returns an error on the request and a **hard** mounted filesystem prints a message and retries the request.

### Read-Write vs. Read-Only

Filesystems that are mounted **rw** (read-write) should use the **hard** option to guard against incomplete writes.

### Interrupting Processes With Pending NFS Requests

The **intr** option (default) allows keyboard interrupts to kill a process that is hung waiting for a response on a hard mounted filesystem. The **nointr** option does not allow keyboard interrupts to kill a process that is hung waiting for a response on a hard mounted filesystem; instead, the operation is guaranteed to complete when the server reboots.

### Transfer size

The number of bytes in a read or write request can be set with the **rsize** and **wsize** options.

### Secure Filesystems

The **secure** option must be given if the server requires secure mounting for the filesystem. Otherwise, all file accesses will be performed by the server with the effective uid specified in the "-anon" entry in its */etc/exports* file, which may mean permission failures on all accesses.

### File Attributes

The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, then the flush time is extended by the time since the last modification (under the assumption that files that changed recently are likely to change soon). There is a minimum and maximum flush time extension for regular files and for directories. Setting **actimeo=*n*** extends flush time by *n* seconds for both regular files and directories.

## UMOUNT OPTIONS

**-h host** Unmount all filesystems listed in */etc/mstab* that are remote-mounted from *host*.

**-t type**

Unmount all filesystems listed in */etc/mstab* that are of a given *type*.

**-a** Attempt to unmount all the filesystems currently mounted (listed in */etc/mstab*). In this case, *fsname* is taken from */etc/mstab*.

**-v** Verbose — *umount* displays a message indicating the filesystem being unmounted.

## EXAMPLES

<code>mount /dev/da0e /usr</code>	mount a local disk
<code>mount -at 4.2</code>	mount all 4.2 filesystems
<code>mount -t nfs serv:/usr/src /usr/src</code>	mount remote filesystem
<code>mount serv:/usr/src /usr/src</code>	same as above
<code>mount -o soft serv:/usr/src /usr/src</code>	same as above but soft mount
<code>mount -p &gt; /etc/fstab</code>	save current mount state

**FILES**

`/etc/mstab` table of mounted filesystems  
`/etc/fstab` table of filesystems mounted at boot

**SEE ALSO**

`mount(2)`, `unmount(2)`, `fstab(5)`, `mstab(5)`, `mountd(8C)`, `nfsd(8)`

**BUGS**

Mounting filesystems full of garbage crashes the system.

NFS mounts directly under `/` cause problems when the server is down, because any process calling `getwd(3)` will result in an attempt to find information about the root of the remote file system. To ameliorate this problem, always mount a file system farther down in the local directory tree and make a symbolic link to the mount point. For example:

```
ln -s /rmt/server/test /test ; mount server:/test /rmt/server/test
```

If the directory on which a filesystem is to be mounted is a symbolic link, the filesystem is mounted on *the directory to which the symbolic link refers*, rather than being mounted on top of the symbolic link itself. Because of this behavior, the above example could also be done as:

```
ln -s /rmt/server/test /test ; mount server:/test /test
```

**NOTES**

When doing a `umount -a` operation, `umount` forks a copy of itself in order to send a broadcast request to all NFS server machines, instructing them to clear their mount entries for this host; see `rmtab(5)`;

NFS is an optional product; for more information, contact your CONVEX sales representative.

## NAME

named - Internet domain name server

## SYNOPSIS

**named** [ **-d** *debuglevel* ] [ **-p** *port#* ] [ **{-b}** *bootfile* ]

## DESCRIPTION

*Named* is the Internet domain name server. See RFC883 for more information on the Internet name-domain system. Without any arguments, *named* will read the default boot file */etc/named.boot*, read any initial data and listen for queries.

Options are:

- d** Print debugging information. A number after the "d" determines the level of messages printed.
- p** Use a different port number. The default is the standard port number as listed in */etc/services*.
- b** Use an alternate boot file. This is optional and allows you to specify a file with a leading dash.

Any additional argument is taken as the name of the boot file. The boot file contains information about where the name server is to get its initial data. If multiple boot files are specified, only the last is used. Lines in the boot file cannot be continued on subsequent lines. The following is a small example:

```

;
;      boot file for name server
;
; directory      /usr/local/domain

; type      domain              source host/file      backup file

cache      .                    .                    root.cache
primary    Berkeley.EDU        berkeley.edu.zone
primary    32.128.IN-ADDR.ARPA  ucbhosts.rev
secondary  CC.Berkeley.EDU     128.32.137.8 128.32.137.3 cc.zone.bak
secondary  6.32.128.IN-ADDR.ARPA 128.32.137.8 128.32.137.3 cc.rev.bak
primary    0.0.127.IN-ADDR.ARPA          localhost.rev
forwarders 10.0.0.78 10.2.0.78

; slave

```

The "directory" line causes the server to change its working directory to the directory specified. This can be important for the correct processing of \$INCLUDE files in primary zone files.

The "cache" line specifies that data in "root.cache" is to be placed in the backup cache. Its main use is to specify data such as locations of root domain servers. This cache is not used during normal operation, but is used as "hints" to find the current root servers. The file "root.cache" is in the same format as "berkeley.edu.zone". There can be more than one "cache" file specified. The cache files are processed in such a way as to preserve the time-to-live's of data dumped out. Data for the root nameservers is kept artificially valid if necessary.

The first "primary" line states that the file "berkeley.edu.zone" contains authoritative data for the "Berkeley.EDU" zone. The file "berkeley.edu.zone" contains data in the master file format described in RFC883. All domain names are relative to the origin, in this case, "Berkeley.EDU" (see below for a more detailed description). The second "primary" line states that the file "ucbhosts.rev" contains authoritative data for the domain "32.128.IN-ADDR.ARPA," which is used to translate addresses in network 128.32 to hostnames. Each master file should begin with

an SOA record for the zone (see below).

The first "secondary" line specifies that all authoritative data under "CC.Berkeley.EDU" is to be transferred from the name server at 128.32.137.8. If the transfer fails it will try 128.32.137.3 and continue trying the addresses, up to 10, listed on this line. The secondary copy is also authoritative for the specified domain. The first non-dotted-quad address on this line will be taken as a filename in which to backup the transferred zone. The name server will load the zone from this backup file if it exists when it boots, providing a complete copy even if the master servers are unreachable. Whenever a new copy of the domain is received by automatic zone transfer from one of the master servers, this file will be updated. The second "secondary" line states that the address-to-hostname mapping for the subnet 128.32.136 should be obtained from the same list of master servers as the previous zone.

The "forwarders" line specifies the addresses of sitewide servers that will accept recursive queries from other servers. If the boot file specifies one or more forwarders, then the server will send all queries for data not in the cache to the forwarders first. Each forwarder will be asked in turn until an answer is returned or the list is exhausted. If no answer is forthcoming from a forwarder, the server will continue as it would have without the forwarders line unless it is in "slave" mode. The forwarding facility is useful to cause a large sitewide cache to be generated on a master, and to reduce traffic over links to outside servers. It can also be used to allow servers to run that do not have access directly to the Internet, but wish to act as though they do.

The "slave" line (shown commented out) is used to put the server in slave mode. In this mode, the server will only make queries to forwarders. This option is normally used on machine that wish to run a server but for physical or administrative reasons cannot be given access to the Internet, but have access to a host that does have access.

The "sortlist" line can be used to indicate networks that are to be preferred over other, unlisted networks. Queries for host addresses from hosts on the same network as the server will receive responses with local network addresses listed first, then addresses on the sort list, then other addresses. This line is only acted on at initial startup. When reloading the nameserver with a SIGHUP, this line will be ignored.

The master file consists of control information and a list of resource records for objects in the zone of the forms:

```
$INCLUDE <filename> <opt_domain>
$ORIGIN <domain>
<domain> <opt_ttl> <opt_class> <type> <resource_record_data>
```

where *domain* is "." for root, "@" for the current origin, or a standard domain name. If *domain* is a standard domain name that does not end with ".", the current origin is appended to the domain. Domain names ending with "." are unmodified. The *opt\_domain* field is used to define an origin for the data in an included file. It is equivalent to placing a \$ORIGIN statement before the first line of the included file. The field is optional. Neither the *opt\_domain* field nor \$ORIGIN statements in the included file modify the current origin for this file. The *opt\_ttl* field is an optional integer number for the time-to-live field. It defaults to zero, meaning the minimum value specified in the SOA record for the zone. The *opt\_class* field is the object address type; currently only one type is supported, IN, for objects connected to the DARPA Internet. The *type* field contains one of the following tokens; the data expected in the *resource\_record\_data* field is in parentheses.

```
A      a host address (dotted quad)
NS     an authoritative name server (domain)
MX     a mail exchanger (domain)
CNAME  the canonical name for an alias (domain)
```

SOA	marks the start of a zone of authority (domain of originating host, domain address of maintainer, a serial number and the following parameters in seconds: refresh, retry, expire and minimum TTL (see RFC883))
MB	a mailbox domain name (domain)
MG	a mail group member (domain)
MR	a mail rename domain name (domain)
NULL	a null resource record (no format or data)
WKS	a well know service description (not implemented yet)
PTR	a domain name pointer (domain)
HINFO	host information (cpu_type OS_type)
MINFO	mailbox or mail list information (request_domain error_domain)

Resource records normally end at the end of a line, but may be continued across lines between opening and closing parentheses. Comments are introduced by semicolons and continue to the end of the line.

Each master zone file should begin with an SOA record for the zone. An example SOA record is as follows:

```
@      IN      SOA      ucbvax.Berkeley.EDU. rwh.ucbvax.Berkeley.EDU. (
                                2.89      ; serial
                                10800     ; refresh
                                3600      ; retry
                                3600000   ; expire
                                86400     ; minimum
```

The SOA lists a serial number, which should be changed each time the master file is changed. Secondary servers check the serial number at intervals specified by the refresh time in seconds; if the serial number changes, a zone transfer will be done to load the new data. If a master server cannot be contacted when a refresh is due, the retry time specifies the interval at which refreshes should be attempted until successful. If a master server cannot be contacted within the interval given by the expire time, all data from the zone is discarded by secondary servers. The minimum value is the time-to-live used by records in the file with no explicit time-to-live value.

#### NOTES

If the empty file `/etc/use_nameserver` does not exist, *named* will not be used. Be sure to use the *touch* command to create this file when working with *named*.

The boot file directives "domain" and "suffixes" have been obsoleted by a more useful resolver based implementation of suffixing for partially qualified domain names. The prior mechanisms could fail under a number of situations, especially when then local nameserver did not have complete information.

The following signals have the specified effect when sent to the server process using the *kill(1)* command.

#### SIGHUP

Causes server to read `named.boot` and reload database.

#### SIGINT

Dumps current data base and cache to `/usr/tmp/named_dump.db`

#### SIGIOT

Dumps statistics data into `/usr/tmp/named.stats` if the server is compiled `-DSTATS`. Statistics data is appended to the file.

**SIGSYS**

Dumps the profiling data in /usr/tmp if the server is compiled with profiling (server forks, chdirs and exits).

**SIGTERM**

Dumps the primary and secondary database files. Used to save modified data on shut-down if the server is compiled with dynamic updating enabled.

**SIGUSR1**

Turns on debugging; each SIGUSR1 increments debug level. (SIGEMT on older systems without SIGUSR1)

**SIGUSR2**

Turns off debugging completely. (SIGFPE on older systems without SIGUSR2)

**FILES**

/etc/named.boot	name server configuration boot file
/etc/named.pid	the process id
/usr/tmp/named.run	debug output
/usr/tmp/named_dump.db	dump of the name server database
/usr/tmp/named.stats	nameserver statistics data
/usr/lib/conf/bind	directory containing RFC's, other information
/etc/use_nameserver	If this empty file does not exist named will not be used.

**SEE ALSO**

kill(1), gethostbyname(3N), signal(3c), resolver(3), resolver(5), hostname(7), RFC882, RFC883, RFC973, RFC974, *Name Server Operations Guide for BIND*

**NAME**

`ncheck` - generate names from *i*-numbers

**SYNOPSIS**

```
/etc/ncheck [-i numbers] [-a] [-s] [-h hash_table_size] [filesystem]
```

**DESCRIPTION**

Note for most normal file system maintenance, the function of `ncheck` is subsumed by `fsck(8)`.

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. The `-h` option allows for a user specified *hash\_table\_size*. The default is 32719, which is adequate for striped *file systems*.

A *filesystem* may be specified.

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

**SEE ALSO**

`sort(1)`, `dcheck(8)`, `fsck(8)`, `icheck(8)`

**DIAGNOSTICS**

When the filesystem structure is improper, `??` denotes the 'parent' of a parentless file, and a path-name beginning with `...` denotes a loop.

## NAME

**newfs** - construct a new file system

## SYNOPSIS

**/etc/newfs** [ **-v** | **-n** ] [ *mkfs\_options* ] *special disktype*

## DESCRIPTION

*newfs* is a "friendly" front-end to the *mkfs(8)* program. *newfs* will look up the type of disk on which a file system is being created in the disk description file */etc/disktab*, calculate the appropriate parameters to use in calling *mkfs*, then build the file system by forking *mkfs*.

If the **-v** option is supplied, *newfs* will print out its actions, including the parameters passed to *mkfs*. The **-n** option will also cause *newfs* to display the parameters that would be passed to *mkfs*, but the actual call to *mkfs* is suppressed.

Options which may be used to override default parameters passed to *mkfs* are:

**-s size** The size of the file system in physical sectors.

**-b block-size**

The block size of the file system in bytes, e.g. 64k or 65536.

**-f frag-size**

The fragment size of the file system in bytes, e.g. 8k or 8192.

**-M maxcontig**

The maximum number of blocks that may be allocated sequentially. The default value used is 1, since UNIX drivers are not capable of scheduling multi-block transfers.

**-R rotdelay**

The minimum number of milliseconds to initiate another disk transfer on the same cylinder. It is used in determining the rotationally optimal layout for disk blocks within a file. The default used is 8 ms.

**-N #sectors/track**

The number of physical sectors per track on the disk.

**-t #tracks/cylinder**

The number of tracks per cylinder on the disk.

**-c #cylinders/group**

The number of cylinders per cylinder group in a file system. The default is to make an estimate based on the file system size that will allow for enough inodes (as inodes are allocated on a cylinder group basis) without creating excessive overhead.

**-m free space %**

The percentage of space reserved from normal users; the minimum free space threshold. The default value used is 10%.

**-r revolutions/minute**

The speed of the disk in revolutions per minute (normally 3600).

**-i number of bytes per inode**

This specifies the density of inodes in the file system. The default is to create an inode for each 2048 bytes of data space. If fewer inodes are desired, a larger number should be used; to create more inodes a smaller number should be given. The **-I** option could also be tried.

**-I number of inodes desired**

An attempt will be made by *newfs* to set up a file system with the desired number of inodes. However, the number specified most likely will not be the exact number of inodes created, because of the other requirements and restrictions that affect the

creation of an "optimal" file system on the specified device.

**-E erase pattern**

Causes the entire volume should be overwritten with the specified pattern before the file system is initialized. The pattern is a 32 bit hex number.

**-S spare sectors/cylinder**

The number of spare physical sectors per cylinder. Required by the IDC.

Note that the *block-size*, *frag-size*, and *bytes-per-inode* arguments can end with a "k"; e.g. **-b 8k** indicates a block size of 8192 bytes. The argument *special* refers to a special file used to access a disk partition, such as */dev/rda0h*. The last argument, *disktype*, refers to a type of disk whose entry can be found in the */etc/disktab* file, such as *dkd-001* or *eagle*.

**FILES**

*/etc/disktab* for disk geometry and file system partition information  
*/etc/mkfs* to actually build the file system

**SEE ALSO**

*disktab(5)*, *fs(5)*, *fsck(8)*, *fsrand(8)*, *mkfs(8)*, *newst(8)*

McKusick, Joy, Leffler; "A Fast File System for Unix", Computer Systems Research Group, Dept of EECS, Berkeley, CA 94720; TR #7, September 1982.

**BUGS**

Should figure out the type of the disk without the user's help.

## NAME

**newst** - make a new striped filesystem

## SYNOPSIS

**newst** [ *options* ] *stripedev diskdev1 type1 diskdev2 type2...*

## DESCRIPTION

**newst** is a program for making "striped filesystems". A striped filesystem is a regular UNIX filesystem which uses a striped disk partition. A striped disk partition is a logical disk partition which is interleaved over several physical disk partitions. Striped partitions are used to take advantage of performance improvements made possible by the parallel operation of the several disk arms which make up the striped disk partition. UNIX filesystems can be mounted on striped disk partitions just as with "normal" disk partitions.

**newst** has three basic jobs: (1) constructing a stripe descriptor based on the characteristics of the different partitions to be combined, (2) loading the stripe descriptor into a table in the UNIX kernel using *putst*(8), and (3) making a UNIX filesystem on the stripe partition using *mkfs*(8). After running **newst**, the new striped filesystem can be checked with *fsck*(8) and mounted with *mount*(8).

The arguments to **newst** consist of a striped-disk device name ( */dev/rstX* ) and a list of pairs of raw disk partition names and device types. For example, */dev/rda2h* and *dkd-001*, respectively. Each name/type pair specifies a disk partition which will become a part of the "striped disk partition". The device type is identical to the type specified for *newfs*(8) in the file */etc/disktab* ( see *disktab*(5) ).

**newst** creates a stripe descriptor entry and adds the entry to the stripe descriptor database file */etc/stripecap*.

The switch options may be used:

- v** Verbose. Prints out the generated stripecap entry.
- n** Nochange. Prints the actions which would otherwise be taken, but does not update */etc/stripecap*, and does not invoke *putst*(8) or *mkfs*(8).
- e** Entry. Create the stripecap entry only; don't invoke *putst*(8) or *mkfs*(8).
- b *block-size***  
Specifies the filesystem block size to be *block-size* bytes, e.g. 64k or 65536.
- f *frag-size***  
Specifies the filesystem fragment size to be *frag-size* bytes, e.g. 8k or 8192.
- m *mazcontig***  
Specifies a value for the filesystem *mazcontig* parameter.
- r *rotdelay***  
Specifies a value for the filesystem *rotdelay* parameter.
- s # *sectors/track***  
Specifies the number of physical disk sectors per track.
- t # *tracks/cylinder***  
Specifies the number of disk tracks per cylinder.
- c # *cylinders/group***  
The number of cylinders per cylinder group in the file system. The default is to make an estimate based on the file system size that will allow for enough inodes (as inodes are allocated on a cylinder group basis) without creating excessive overhead.
- I *number of inodes desired***  
An attempt will be made by **newst** to set up a striped file system with the desired number of inodes. However, the number specified most likely will not be the exact

number of inodes created, because of the other requirements and restrictions that affect the creation of an "optimal" file system on the specified devices.

**-E erase pattern**

Causes the entire volume should be overwritten with the specified pattern before the file system is initialized. The pattern is a 32 bit hex number.

**-S spare sectors/cylinder**

The number of spare physical sectors per cylinder. Required by the IDC.

Note that the *block-size* and *frag-size* arguments can end with a "k", e.g. "-b 8k" to indicate 8192-byte blocks. Disk partition size information is obtained from */etc/disktab* based on the disk types supplied in the argument list. Disk geometric information, such as disk block size, is obtained either from the switches, or else from the */etc/disktab* entry of the first disk device given in the argument list (see *disktab(5)*).

*newst* uses a set of heuristics to determine how to lay out logical stripe sections on the physical devices. These heuristics are derived from performance measurements using stripe filesystems. The user need not be aware of them except that filesystem bandwidth is optimized if only one partition per physical disk drive is used in a striped filesystem. (It is also currently a requirement for optimum performance that a separate controller be used for each disk drive, and a separate Multibus chassis for each controller).

**EXAMPLE**

**newst /dev/rst0 da4a dkd-001 da5a dkd-001 da6h dkd-001**

The above command creates a three disk striped partition, using the disk partitions */dev/rda4a*, */dev/rda5a*, and */dev/rda6h*, each of which is of type *dkd-001* as found in */etc/disktab*.

**FILES**

<i>/dev/rda??</i>	Raw devices for the disk driver
<i>/dev/rst?</i>	Raw stripe device
<i>/etc/stripecap</i>	Permanent home of the stripe descriptors
<i>/etc/disktab</i>	For disk geometry and file system partition information
<i>/etc/mkfs</i>	To actually build the file system

**SEE ALSO**

*putst(8)*, *getst(8)*, *mkfs(8)*, *fsirand(8)*, *newfs(8)*, *stripecap(5)*, *st(4)*

**NAME**

nfaccess – add access rights to a set of Notesfiles

**SYNOPSIS**

```
nfaccess access-right topic [ topic ... ]
```

**DESCRIPTION**

**nfaccess** simplifies the task of adding an access-right to many notesfiles. The function is somewhat analogous to that of *chmod*(1).

The *access-right* specifies a user, group or system and the permissions to be granted. The format is:

```
<access-right> ::= [<type>:]name=<mode>
<type>          ::= {User, user, Group, group, System, system}
<mode>          ::= {d, r, w, a, n}+
```

The *type* specification can be omitted; when it is omitted, the name is assumed to be a user. The *mode* is additive. A mode of “rw” specifies read and write. The “n” mode specifies null access. The “d” mode specifies director privileges. The “a” mode specifies answer, meaning that the user can respond to notes, but not write them.

The new access right is inserted in each specified notesfiles access list. If the user/group/system already has an entry, the old entry is replaced with the new entry.

All users are allowed to run this program. The changes are only applied to notesfiles for which the executing user is a director.

This program lives in the notesfile utility directory, typically “/usr/spool/notes/.utilities”.

To automatically add specific access-rights to newly created notesfiles, the file “/usr/spool/notes/.utilities/access-template” is useful. If it exists, the file contains lines of access-rights which are added to each notesfile as it is created. If several people share the administration of the notesfile system, each can be added to the access-lists of newly created notesfiles by placing appropriate lines in this file.

**EXAMPLES**

```
nfaccess essick=drw /usr/spool/notes/*
nfaccess group:srg=rw this that other
nfaccess user:smith=rw mynotes
```

In the first example, user “essick” is given director/read/write access to all the notesfiles in /usr/spool/notes. Any permissions (or restrictions) he might have had before are overridden. The second example gives group “srg” read/write access to notesfiles “this”, “that” and “other”. In the final example, user “smith” is given read/write access to the notesfile “mynotes”.

**BUGS**

Entries can not be removed from the access list with this program.

Entries can not merely be augmented (“just add write permission”) with this program.

**FILES**

/etc/passwd	for the users name
/etc/group	for the users group
/usr/spool/notes	the default notesfile data base

**SEE ALSO**

mknf(8), notes(1), notesadm(8), nfcomment(3)  
 “Notesfile Reference Manual” in the *CONVEX UNIX Tutorial Papers*

## NAME

*nfarchive* – archive notesfiles

## SYNOPSIS

**nfarchive** [ **-#** ] [ **-d** ] [ **-m-** or **-m+** ] [ **-w#** ] [ **-f** file ] topic [ ... ]

## DESCRIPTION

*Nfarchive* is used to expire notes that have not been modified in a certain amount of time. Archives are stored in “archive notesfiles”.

The **-#** parameter is the number of days a notestring must be idle (no new responses) before being eligible for archival.

Expired notestrings are either deleted or placed in an archive. The **-d** parameter tells *nfarchive* to delete expired notestrings. If unselected, the expired notestrings are placed in an archive.

The **-m+** option specifies that only notes marked with a director message are eligible for expiration. **-m-** specifies that only notes without a director message are eligible for expiration. By default, the expiration algorithm is indifferent to a note’s director message status.

The **-w#** option specified the working set size for the expired notesfiles. The specified number represents the minimum number of notes to leave in the notesfile.

The expiration threshold, working set size, expiration action, and director message requirements can all be specified as a director’s option in each notesfile. Specific values override what is specified on the *nfarchive* command line. A default value specifies using the value specified on the command line.

The **-f** parameter specifies a file containing a list of notesfiles to archive. Notesfiles can also be specified on the command line.

By default, the archive of notesfile */usr/spool/notes/somenotes* is in */usr/spool/oldnotes/somenotes*. The archive of */some/other/place/somenotes* also defaults to */usr/spool/oldnotes/somenotes*. To prevent collisions of this nature, mapping between active and archive notesfiles is implemented. The file */usr/spool/notes/.utilities/net.alias/Archive-into* contains lines of the form:

```
active-notesfile:archive notesfile
```

Lines in this file beginning with ‘#’ are comments. Notesfiles without an entry in this file are archived into the */usr/spool/oldnotes* directory with the appropriate last component.

When initially created, an archive notesfile has an access list matching its active counterpart. Currently, only directors are allowed to write in an archive notesfile. *Nfarchive* refuses to archive an archive notesfile.

The user ‘notes’ and the director(s) of a notesfile are the only users that are permitted to run *nfarchive* on that notesfile.

## FILES

<i>/usr/spool/notes/.utilities</i>	where this programs lives.
<i>/usr/spool/notes/.utilities/net.alias/Archive-into</i>	maps active notesfiles into their archives.
<i>/usr/spool/notes</i>	Default notes data base
<i>/usr/spool/oldnotes</i>	Default archive directory

## SEE ALSO

notes(1), notesadm(8)  
 “Notesfile Reference Manual” in the *ConvexOS Tutorial Papers*

## AUTHORS

Ray Essick

Rob Kolstad

**NAME**

**nfdump**, **nload** – notesfile dump/load programs

**SYNOPSIS**

**nfdump** notesfile result

**nload** [ **-D**directory ] notesfile

**DESCRIPTION**

*nfdump* and *nload* are used to convert notesfile data base formats.

*nfdump* converts the specified notesfile to a portable ASCII format and places it in the file specified by the *result* argument. If *result* is a single minus sign (“-”), the output from *nfdump* is sent to standard output.

*nload* is used on the output from *nfdump* to create a new notesfile. The **-D** option specifies an alternate base directory for the notesfile. If unspecified, this defaults to “/usr/spool/notes”. *nload* reads standard input for the ASCII representation of the notesfile.

Typical use of these two programs occurs when converting an existing notesfile data base to a new format. The *nfdump* program should be compiled with the older structure definitions while *nload* is compiled with the newer structures. The data base can then be converted with a shell script of the following nature:

```
mkdir .OLD
mv * .OLD
for i in `ls .OLD`
do
    echo $i start
    nfdump $i - | nload -D/usr/spool/newnotes $i
done
    echo $i done
echo ALL DONE
rm -rf .OLD
```

This assumes that the old data base is in “/usr/spool/notes” and the new data base is to be placed in “/usr/spool/newnotes”. After the conversion is complete, one can move the old data base from “/usr/spool/notes” to “/usr/spool/oldformat” and the new data base from “/usr/spool/newnotes” to “/usr/spool/notes”.

**SEE ALSO**

notes(1)

“Notesfile Reference Manual” in the *CONVEX UNIX Tutorial Papers*

**NAME**

`nfmail` – accept mail for a notesfile

**SYNOPSIS**

`nfmail` [ `-s` ] [ `-m` mailrc ] topic

**DESCRIPTION**

*Nfmail* is a mail-receiving program which takes incoming mail, parses subject and author information, and places the letter in a notesfile. Replies, marked by a “Re:” prefix in the Subject line, are placed in the notesfile as responses if a basenote with the appropriate title can be found.

The `-s` option tells *nfmail* to strip header lines from the letter before placing it in the notesfile. Normally, all header information is retained. The “Subject”, and “From” header lines are never removed from the letter. The file `/usr/lib/Mail.rc` defines default mail reading characteristics on many BSD systems. This file can contain lists of header lines to be ignored when presenting messages to users. *Nfmail* reads this file to determine which header lines are normally ignored. If this file is missing, *nfmail* doesn’t strip any header lines.

The `-m` option specifies further files, typically “.mailrc” files, to search for header lines to ignore. More than one `-m` option can appear on the command line.

*Nfmail* usually appears as a mail alias in the file `/usr/lib/aliases` in lines such as:

```
problems:      "/usr/spool/notes/.utilities/nfmail -s problems"
```

**BUGS**

The dependence on `/usr/lib/Mail.rc` for lists of headers to be ignored should be cleaned up. Perhaps something where absence of the file means to strip all headers except the Subject and From lines.

**FILES**

<code>/usr/spool/notes/.utilities/nfmail</code>	where this program lives.
<code>/usr/lib/aliases</code>	Mail aliases (BSD UNIX)
<code>/usr/lib/Mail.rc</code>	Mail configuration templates

**SEE ALSO**

`notes(1)`, `notesadm(8)`, `mail(1)`  
 “Notesfile Reference Manual” in the *ConvexOS Tutorial Papers*

## NAME

nfxmit, nfrcv – notesfile networking programs

## SYNOPSIS

```
nfxmit -dsite [ -r ] [ -i ] [ -p ] [ -a ] [ -t date_spec ] [ -f file ] nf [ nf2 ... ]
nfrcv topic fromsystem
```

## DESCRIPTION

*nfxmit* and *nfrcv* implement networking of notesfiles. The *nfxmit* program packages eligible new articles and sends them to a remote system. *nfrcv* is the receiving end of this team.

*nfxmit* sends the specified notesfiles to the *site* specified with the **-d** option. The **-r** option specifies that a request should be queued for the remote site to transmit updates from its copies of the notesfiles sent. The **-r** option is used only if the other site does not automatically queue updates of the notesfile.

The **-i** option indicates that *nfxmit* is to send articles to the destination that it normally would not send. These articles are those that were written there or were received at the local site from that particular site. The **-i** option is separate from the timestamp functions. Articles are first selected on the basis of the time received and criteria for having passed the destination system are applied later.

The **-p** option indicates that *nfxmit* is to send director's message and title changes of articles. If the director's message or title has changed since the last transmission to *site*, then the new title text and director's message state, on or off, are propagated. Note, both the local and remote notesfile must have *Propagate Change* set in the director option page of their respective notesfiles in order for the propagation to be completed.

The **-a** option indicates that *nfxmit* is to send articles which have originated from *news* in addition to *notes*-generated articles. Normally *nfxmit* will only send articles which are originated from within the *notes* system.

A timestamp of the last transmission of each notesfile to each system is maintained. This is used for determining the notes to send. If a different timestamp is desired, use the **-t** option. The supplied timestamp is used for this transfer and the stored timestamp is updated.

Specify **-f** *myfile* on the command line to have *nfxmit* read *myfile* for a list of notesfiles to be sent. This is useful if the number of notesfiles is too numerous to list on a single command line. The shell meta-characters \*, ?, [, and ] are recognized in both the *topic* parameter and the entries in *myfile*.

*nfxmit* uses *uux*(1) to invoke *nfrcv* on the remote system in order to process the incoming notes.

## FILES

/usr/spool/notes/.utilities/net.how	Specifies connection methods between systems.
/usr/spool/notes/.utilities/net.alias	Directory containing mapping of local and remote notesfile names.
/usr/spool/notes/.sequencer/Sy: <i>sysname</i>	Sequencing timestamps for <i>sysname</i> .
/usr/spool/notes/.sequencer/Sy: <i>sysname</i> ,p	Director's message and title change propagation sequencing timestamps for <i>sysname</i> .

## SEE ALSO

notes(1), notesadm(8), nfcomment(3), uucp(1c)  
 "Notesfile Reference Manual" in the *ConvexOS Tutorial Papers*

**NAME**

nu - add a new user to the system

**SYNOPSIS**

nu [ -f *batchfile* ] [ -n *nurcfile* ]

**DESCRIPTION**

nu adds a new user to the system, creating directories and setting up startup files as necessary. nu can be entirely interactive (if invoked without the -f flag), prompting the user for any and all information necessary to complete the user addition.

Many questions will have default values that may be used. These defaults will be displayed in square brackets, along with the question. It is always necessary to enter a login name - there is no default for this value.

Default values to override those hardwired into the code itself can be defined in an *nurc* file, either */etc/nurc* or an alternate file given with the -n flag. The format of an *nurc* file is simply lines of the form *fieldname:fieldvalue* to set a default value, or *fieldname* to "turn on" a boolean-type field, such as enabling password type or age restrictions..

The possible fields which can be defined in an *nurc* file are as follows, given with their hardwired defaults, if any:

Name	Default	Description
uid	(see below)	user ID
gid	EMPTY	group ID
group	"staff"	group name (used only if gid field not given)
directory	"/mnt"	path under which home directory will be placed
protection	"0755"	home directory protection
shell	"/bin/csh"	login shell
password	(see below)	new user's initial password
username	"username"	user's full name (for the <i>finger</i> command)
office	"office"	user's office (for the <i>finger</i> command)
extension	"extension"	user's work extension
homephone	"homephone"	user's home phone number
minwks	"1"	minimum # of weeks for password aging
maxwks	"52"	maximum # of weeks for password aging
diskquota	"6000,8000,1200,1500"	<i>blksoft</i> , <i>blkhard</i> , <i>inodesoft</i> , <i>inodehard</i> quotas
skeleton	"/usr/skel"	directory containing files to copy into home directory
homedir	< <i>directory</i> >/< <i>login</i> >	home directory to create for this user
sgroup	"staff"	parent share group of new user
shares	"100"	number of shares assigned to new user

Boolean	Default	Description
typed	OFF	should this user have password typing restrictions?
aged	OFF	should this user have password aging restrictions?
quota	OFF	should home directory file system have quotas initialized?
nouidfile	OFF	ignore the contents of <i>/etc/uidcount</i> ?
newsgroup	OFF	is this login a new share scheduling group?
notshared	OFF	is this login a not-shared scheduling group?

The default uid will be the value contained in the */etc/uidcount* file, unless the *nouidfile* boolean is set. If this is the case, the default will be the number one greater than the highest uid already in use in the password file.

Values for uids must be greater than or equal to zero and less than 32766 (the values 32766 and 32767 are reserved for use with Share; root must always have uid 0.) Values outside this range are illegal.

If zero is specified for the minimum and maximum weeks for password aging, the user will be forced to change his password the next time he logs in (and the "age" will disappear from his entry in the restrictions file). If the minimum is greater than the maximum, only the super-user will be able to change the password. Therefore, when 0 is specified for the aging defaults, password aging is turned off.

In batch mode, the default password will be the user's login name. However, when running *nu* interactively, the *passwd* utility will be invoked, forcing creation of an initial password by the system administrator.

A sample *nurc* file might look like this:

```
shell: /bin/csh
group: swtst
directory: /swtst
protection: 0755
typed
quota
diskquota: 3000, 4096, 1000, 1500
```

The batch input file (given with the *-f* flag) has a slightly different format than the *nurc* file, but the fields that can be defined include all those listed above for *nurc* files. The batch file contains information on any number of new users, one user per line. Each line is a colon-separated list of value specifications for certain variables of the form *fieldname=fieldvalue* to define a specific value, or more simply *fieldname*, to just turn a boolean field "on." There is one additional field that must be set for each new user, namely the *login* field.

The fields which are given for a certain user affect only that user. Before processing each user in the batch file, all fields are re-initialized to the defaults hardwired in or given in an *nurc* file. Fields that will be common to a large group of new users should be placed in an *nurc* file, and only the necessary user-specific fields defined within the batch file.

Given the following example entry in a batch file, it is easy to ascertain the meaning of all of the fields:

```
login=jsmith:username=Julie Smith:office=107A:extension=555:typed
```

*Nu* tries to do reasonable things while it performs its function, like:

1. Only one copy at a time may run.
2. Uses *mkpasswd* to rebuild the password and restrictions databases.
3. The *chfn* and *passwd* utilities do all of the dirty work.
4. Exits with coherent status.
5. Checks for illegal data (like ":") in interactively-input fields.
6. Interrupts are disabled during critical sections of the code.

All files in a specified skeleton directory (*/usr/skel* is the default) are copied into the user's new home directory. Sample shell source files such as *.cshrc*, *.profile*, and *.login* usually appear in this skeleton directory to provide a basis for new users on the system.

The C source for the *nu* program is shipped as part of the standard distribution of utilities, in order to provide for local customization by each site's system administrator.

#### FILES

<i>/usr/skel/*</i>	skeleton initialization files (if alternate <i>skeleton</i> not given)
<i>/etc/nurc</i>	<i>nu</i> default values file (if alternate <i>nurc</i> file not given)

<code>/etc/group</code>	group file
<code>/etc/passwd</code>	password file
<code>/etc/pwrestrict</code>	password restrictions file
<code>/etc/uidcount</code>	next user ID free to use (if <i>nouidfile</i> boolean not set)
<code>/etc/ptmp</code>	password lock file
<code>/etc/rtmp</code>	restrictions lock file
<code>/tmp/nu.lock</code>	<i>nu</i> lock file
<code>/etc/shares</code>	the share data base file
<code>/usr/convex/nu.c</code>	<i>nu</i> C source for local modification

**SEE ALSO**

`chfn(1)`, `passwd(1)`, `pl(1)`, `rates(1)`, `nurc(5)`, `passwd(5)`, `pwrestrict(5)`, `share(5)`, `mkpasswd(8)`, `lim(8)`, `remshent(8)`

**NAME**

*on*, *off* – enable/disable logins on a tty

**SYNOPSIS**

```
on [ -n ] [ -s ] [ ttyname ... ]
off [ -n ] [ -s ] [ -f ] [ ttyname ... ]
```

**DESCRIPTION**

*on* and *off* are used to enable or disable logins, respectively, on specified ttys. These programs cause three events to occur. First, */etc/ttys* is rewritten. Next, the disk containing */etc/ttys* is synchronized using *fsync(2)*. Finally, the *init* program (process 1) is signaled to re-read */etc/ttys* and to appropriately set up the ttys. *on/off* attempts to synchronize operations through a lock file. If this lock file cannot be acquired, a message to this effect is displayed and the process exits. This is not a true error; it just means that another *on/off* was running at the time.

The *ttynames* should be specified as they appear in */etc/ttys*; that is, the full pathname with */dev/* stripped off.

If someone is logged on to a tty when *off* is executed, */etc/ttys* is modified, but *init* is not signaled. This protocol allows the user to finish work without interruption. The **-f** flag forces a logoff.

The **-n** flag causes *on/off* to modify */etc/ttys*, but not to signal *init*.

The **-s** flag causes *init* to be signaled, with no change to */etc/ttys*.

**NOTES**

*off* is actually a link to *on*; the function that is performed is determined by the name under which the program was invoked.

These programs manipulate */etc/ttys* and *init*; they can therefore be executed only by the superuser.

**FILES**

```
/etc/ttys      tty status
/etc/utmp    tty login status
/tmp/off.lock synchronization lock file
```

**EXIT STATUS**

If the operation was successful and no one was logged on the tty, **0** is returned; **1** is returned if someone was logged on. If the operation was not successful, various error codes from *<sysexit.h>* are returned.

**SEE ALSO**

*fsync(2)*, *ttys(5)*, *utmp(5)*, *init(8)*

## NAME

*op* – operator interface tool for giving restricted access to privileged commands

## SYNOPSIS

```
/etc/op mnemonic [ arg ... ]
/etc/op -h [ -u username ] [ mnemonic ]
```

## DESCRIPTION

The *op* tool provides a flexible means for system administrators to grant to any set of trusted users permission, or access, to execute certain root operations without having to give them full superuser privileges. It is a non-interactive command interpreter that places restrictions on which users are allowed to execute which privileged commands.

The functions (or mnemonics) understood by the *op* program are listed in the configurable data file */etc/op.access*, along with the meaning of each mnemonic (an exact UNIX command that will accomplish the desired result) and who is allowed to execute it. The restrictions can be made as tight as each site demands, as determined by the system administrator who customizes the *op.access* file.

The format of the access database file is fully described in *op.access(5)*. In summary, it contains a mapping of mnemonics, or operator functions, to the full pathnames of programs that should be invoked and the arguments that are allowed, if any. The arguments to the executed program can be a combination of literal and variable arguments, and restrictions can be placed on which values are valid substitutions for the variable arguments. The *args* given on the *op* command line are only necessary to specify any variable arguments the mnemonic may need. While the superuser does not have access permissions checked (root can run anything), each *arg*'s validity is verified.

The following set of attributes can also be controlled for each mnemonic by the *op* program:

- the uid to set (root by default)
- the gid to set (not changed, by default)
- the group vector to set (contains only gid, by default)
- the directory to *chdir(2)* to (not changed, by default)
- the root directory to set with *chroot(2)* (not changed, by default)
- the umask to set (022 by default)
- a list of groups allowed to execute this function (none by default)
- a list of users allowed to execute this function (none by default, except the superuser)
- the range of valid arguments for the command (any value per variable argument by default)
- any number of environment variable settings (none by default).

The following options are recognized:

**-h** [ *mnemonic* ]

This option requests help and informs the operator of the commands he or she is allowed to run and how they are to be run. The operator is not allowed to view the access file directly for security reasons. Without an argument, this option will display a list of mnemonics which the invoking user is allowed to execute. If a specific mnemonic is given, and if that mnemonic is defined in the access file and the user has permission to execute it, then a "usage" message is output, describing the valid arguments accepted by that mnemonic and the order in which they should be given (if there are any variable arguments to that mnemonic at all).

**-u** *username*

Use the specified *username* when checking for permission to execute the mnemonics listed in the */etc/op.access* file. This option may only be used by the superuser and in conjunction with the **-h** option.

If the `-h` flag (for help) is not given, the `op` program verifies that the invoking user is allowed to run a particular command, validates any variable arguments, and properly sets up any of the above-specified attributes before executing the associated program. Security measures such as clearing out the vector of group permissions and clearing all but the specified environment variables are taken before the command is executed.

Attempted executions of `op` (whether successful or not) are logged via `syslog(3)`. The information logged includes when and by whom `op` was executed and with which command-line arguments. Invalid users or variables are especially noted. Many messages written to the log file contain the user's login name, which will be contained within square brackets (`[]`).

All `op` logging is done using `syslog`'s `LOG_AUTH` facility. The usage information will be logged with a `syslog` level of `LOG_INFO`, usage errors (such as invalid or wrong number of arguments) use the `LOG_NOTICE` level, an unauthorized user will be noted with the `LOG_WARNING` level, and a failed `execve` of the command will be logged with the `LOG_ERR` level. The log files to which this `syslog` information is written can be customized within the `/etc/syslog.conf` file; see `syslogd(8)` for information on how to configure this file.

#### EXAMPLES

The following line in the `op.access` file gives the user `bruce` and any member of the `opers` group permission to run weekly dumps on any of the named file systems by simply typing `op weekly valid_filesys_name`, where `valid_filesys_name` is one of `/`, `/usr`, or `/mnt`. Because no other attributes are defined for this mnemonic, the defaults are used: the dump program runs as root, but the gid, current working directory, and root directory remain unchanged. The umask is 022, and the group and environment vectors are cleared.

```
weekly      /etc/dump 0Gun $1;      users=bruce groups=opers
                                     $1=/,/usr,/mnt
```

For more examples of the access file format and usage, see the `op.access(5)` manual page.

#### FILES

`/etc/op.access` list of operator mnemonics and restrictions enforced by `op`

#### SEE ALSO

`op.access(5)`, `syslog(3)`, "Operator Interface" chapter in the *CONVEX System Manager's Guide*

#### WARNINGS

It is up to each individual system administrator to assure that the programs listed within the `op.access` file are secure. It is recommended that the access file not contain shell scripts that run as root. Also, caution should be taken in including any interactive programs in the access database.

**NAME**

opreq\_daemon - opreq daemon

**SYNOPSIS**

`/usr/lib/opreq/opreq_daemon [ -unmount ]`

**DESCRIPTION**

*opreq\_daemon* is the main daemon for the opreq system. It is normally started from */etc/rc.local*. It must be started before the *tpdaemon(8)*.

**OPTIONS**

**unmount** Specifying this option will cause unmount requests to be deleted automatically. Normally they must be removed by the operator using the *opreq(1)* program.

**FILES**

<code>/usr/lib/opreq/share*</code>	Shared memory files
<code>/usr/lib/opreq/opreq_share.lockfile</code>	Shared memory lock file
<code>/usr/lib/opreq/opreq_daemon.lockfile</code>	opreq_daemon lock file

**SEE ALSO**

*opreq(1)*, *tpdaemon(8)*

## NAME

`pac` - printer/plotter accounting information

## SYNOPSIS

`/etc/pac [ -p price ] [ -s ] [ -r ] [ -c ] [ -g ] [ -a ] [ -G ] [ -A ] [ -U ] [ acct_file ]`

## DESCRIPTION

`pac` reads the printer/plotter accounting file, accumulating the number of pages (the usual case) or feet (for raster devices) of paper consumed by each user, and printing out how much each user consumed in pages or feet and dollars. Usually, statistics are reported for every user who has used any paper. The `-g` and the `-a` options report the printer statistics by group and activity, respectively. These preceding options specify the primary field used for reporting. The `-G`, `-A`, and `-U` options include group, activity, and user information as well as information specified by the primary field. These uppercase options include group, activity, and user information in the report, respectively. For example,

```
/etc/pac -g lpd-acct
```

will report printer statistics by group;

```
/etc/pac -g -A lpd-acct
```

will report printer statistics by group and activity;

```
/etc/pac -G -A lpd-acct
```

will report printer statistics by user, group, and activity.*n*

The `-p` flag causes the value *price* to be used for the cost in dollars instead of the default value of 0.02.

The `-c` flag causes the output to be sorted by cost; without this flag, the output is sorted by the first field. With no other flags, the first field defaults to "user name".

The `-r` flag reverses the sorting order.

The `-s` flag causes the accounting information to be summarized on the summary accounting file; this summarization is necessary since on a busy system, the accounting file can grow by several lines per day. Summarization updates data in the summary accounting file and causes the raw accounting file to be truncated.

## FILES

<code>/usr/adm/?acct</code>	raw accounting files
<code>/usr/adm/?_sum</code>	summary accounting files

## SEE ALSO

`idtoname(1)`, `lpd-acct(5)`, `sumscripts(8)`,  
 "Accounting" chapter in the *CONVEX System Manager's Guide*.

## NOTES

`pac` is a shell script. If the output of `pac` is not acceptable, local modifications can be made. The script can be modified in, say, `pac.local` to produce acceptable output.

## BUGS

The relationship between the computed price and reality is as yet unknown.

**NAME**

ping - send ICMP ECHO\_REQUEST packets to network hosts

**SYNOPSIS**

```
/usr/etc/ping [ -r ] [ -v ] host [ packetsize ] [ count ]
```

**DESCRIPTION**

The DARPA Internet is a large and complex aggregation of network hardware, connected together by gateways. Tracking a single-point hardware or software failure can often be difficult. *Ping* utilizes the ICMP protocol's mandatory ECHO\_REQUEST datagram to elicit an ICMP ECHO\_RESPONSE from a host or gateway. ECHO\_REQUEST datagrams ("pings") have an IP and ICMP header, followed by a **struct timeval**, and then an arbitrary number of "pad" bytes used to fill out the packet. Default datagram length is 64 bytes, but this may be changed using the command-line option. Other options are:

- r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by *routed*(8C)).
- v Verbose output. ICMP packets other than ECHO\_RESPONSE that are received are listed.

When using *ping* for fault isolation, it should first be run on the local host, to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be "pinged". *Ping* sends one datagram per second, and prints one line of output for every ECHO\_RESPONSE returned. No output is produced if there is no response. If an optional *count* is given, only that number of requests is sent. Round-trip times and packet loss statistics are computed. When all responses have been received or the program times out (with a *count* specified), or if the program is terminated with a SIGINT, a brief summary is displayed.

This program is intended for use in network testing, measurement and management. It should be used primarily for manual fault isolation. Because of the load it could impose on the network, it is unwise to use *ping* during normal operations or from automated scripts.

**SEE ALSO**

netstat(1c), ifconfig(8C)

**NOTES**

*Ping* is an optional product; for more information, contact your CONVEX sales representative.

**NAME**

*preen* - run *fsck* in parallel over normally mounted disks

**SYNOPSIS**

*/etc/preen*

**DESCRIPTION**

*Preen* invokes *fsck* to check disk partitions in a highly optimal manner, much in the same way that the **-p** option in *fsck* works. It differs from *fsck* in that it uses the base names of the disks to determine which are the same physical drive, and actually tries to keep all drives busy. *Fsck* -*p* merely runs groups of "passes", and with odd configurations tends to leave the arms idle a lot.

Status returned by *preen* matches (as closely as possible) that of *fsck*.

One of the biggest advantages of using *preen* is that the system administrator no longer needs to worry about configuring "pass numbers" in the file system table.

**SEE ALSO**

*fsck*(8)

**FILES**

*/etc/fstab*  
*/etc/rc*

## NAME

pstat - print system facts

## SYNOPSIS

```
/etc/pstat -aivpfstT [-Ppid] [-Cname] [-S [uid]] [system] [corefile]
```

## DESCRIPTION

*pstat* interprets the contents of certain system tables. If *corefile* is given, the tables are sought there, otherwise in */dev/mem*. The required namelist is taken from */vmunix* unless *system* is specified. Options are

**-a** Under **-p**, describe all process slots rather than just active ones.

**-i** Print the inode table including the associated vnode entries with these headings:

ILOC The core location of this table entry.

IFLAG Miscellaneous inode state variables encoded thus:

A inode access time must be corrected

C inode has been changed

L inode is locked

R inode is being referenced

T inode contains a pure text prototype

U update time (*fs(5)*) must be corrected

W wanted by another process (L flag is on)

## IDevice

Major and minor device number of file system in which this inode resides

INO I-number within this device

MODE Mode bits in octal, 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.

## VFLAG

Miscellaneous vnode state variables encoded thus:

R root of its file system

S shared lock applied

E exclusive lock applied

O a virtual memory object is associated with this vnode

T vnode is a pure text prototype

Z process is waiting for a shared or exclusive lock

CNT Number of open file table entries for this inode.

SHC Reference count of shared locks on the vnode.

EXC Reference count of exclusive locks on the vnode (this may be > 1 if, for example, a file descriptor is inherited across a fork).

TYPE Vnode file type, either VCON (no type), VREG (regular), VDIR (directory), VBLK (block device), VCHR (character device), VLNK (symbolic link), VSOC (socket), or VBAD (bad).

**-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 runnable

2 being created

3 being terminated

4 awaiting an event

6 stopped awaiting a signal

FLAG	Miscellaneous state variables, or-ed together (hexadecimal):
00000001	process is loaded
00000002	a system process (scheduler or page-out daemon)
00000008	user area and page tables are loaded
00000010	process is being traced
00000020	used in tracing
00000040	process is locked in core
00000200	restore old sigmask after taking signal
00000400	process is exiting
00000800	doing physical i/o
00001000	process resulted from a <i>vfork(2)</i> which is not yet complete
00002000	another flag for <i>vfork(2)</i>
00004000	process has no virtual memory, as it is a parent in the context of <i>vfork(2)</i>
00008000	process was stopped while sleeping
00010000	process wants all threads to stop
00020000	process wants all threads to join
00040000	process is in a sleep which will timeout.
00100000	process used 4.1BSD compatibility mode signal primitives, no system calls will restart.
00200000	process is owed a profiling tick.
00800000	process is a login process.
02000000	process doing asynchronous i/o
04000000	waiting on async i/o completion
10000000	track lock
20000000	parent inherits signals
40000000	release process on debug close
80000000	record-locking has been done
PRI	Scheduling priority, see <i>setpriority(2)</i> .
SIG	Signals received (signals 1-32 coded in bits 0-31),
UID	Real user ID.
SLP	Amount of time process has been blocked.
TIM	Time resident in seconds; times over 65535 coded as 65535.
CPU	Weighted integral of CPU time, for scheduler.
NI	Nice level, see <i>setpriority(2)</i> .
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.
UADDR	The virtual address of the processes user structure.
RSS	Resident set size - the number of physical page frames allocated to this process.
SRSS	RSS on the swap device (0 if never swapped).
SIZE	Virtual size of process image (data+stack) in multiples of 4096 bytes.
WCHAN	Wait channel number of a waiting process.

-v Prints information about each process's virtual memory. First the LOC, S, FLAG, PRI, PID, PPID, UADDR, RSS, SRSS, SIZE, and WCHAN fields are printed from the process table (as described above). Then information from the process's vspace structure is printed, followed by a line for each region attached to the vspace.

For each vspace, the following information is printed:

- LOC** The address of the vspace structure.
- FLAGS** Flags about the vspace structure, as follows:
- L** vspace is locked
  - S** vspace is locked shared
  - E** vspace is locked exclusively
  - W** vspace is wanted by another
  - R** regions virtually present only
  - N** vspace is nonswappable
  - P** vspace is prepaged
- USE** The use count for the vspace.
- SHARE**  
The number of threads holding shared locks on the vspace.
- NREG** The number of regions attached to the vspace.
- FLOW** Front hand of the page daemon clock that scans vspace for unreferenced pages.
- BLOW** Back hand of the page daemon clock that scans vspace for unreferenced pages.
- For each region, the following information is printed:
- LOC** The address of the region structure.
- FLAGS** Various flags about the region structure, as follows:
- B** if the region is marked **BUSY**
  - W** if the region is marked **WANTED**
  - I** if the region immediately migrates pages to the backing object
- PROT** The protection property of the region, as follows:
- r** the pages are readable
  - w** the pages are writable
  - x** the pages are executable
- SHARE**  
The region sharing parameters (see /usr/include/sys/mman.h), as follows:
- F** **MAP\_FILE** is set
  - A** **MAP\_ANON** is set
  - D** **MAP\_DEVICE** is set
  - P** **MAP\_PRIVATE** is set
  - I** **MAP\_INHERIT** is set
  - d** the region may grow downward
  - u** the region may grow upward
  - a** the region grows automatically without system call intervention
  - s** the region is marked as being sequentially accessed
- VLOW** The lowest virtual page number in the region.
- VSIZE** The number of virtual pages in the region.
- RSS** The resident set size of the region.
- VLIMIT**  
The limit on region growth in pages.
- OBJECT**  
The address of the paging object.  
The address of the backing object.
- f** Print the open file table with these headings:
- LOC** The core location of this table entry.
- TYPE** The type of object the file table entry points to.
- FLG** Miscellaneous state variables encoded thus:
- R** open for reading
  - W** open for writing
  - A** open for appending

- S        shared lock present  
 X        exclusive lock present  
 I        signal pgrp when data ready  
 CNT     Number of processes that know this open file.  
 DATA   The location of the vnode table entry or socket for this file.  
 OFFSET  
        The file offset (see *lseek(2)*), or the core address of the associated socket structure.
- s       Prints several pieces of information about swap space usage. The first line shows the number of kilobytes used and free as well as the number of kilobytes that have been lost from the resource map. Starting on the second line, a list of various size ranges of swap areas is shown, and the number of kilobytes of free space in each range is printed. Below that are the numbers of swap space allocations which succeeded, were truncated, and failed since UNIX was booted. Finally, the number of kilobytes of memory used to keep track of swap space allocations is printed.
- t       Prints information about each thread of a process. This information is implementation dependent. Consult the include file `<sys/thread.h>` for more information.
- T       prints the number of used and free slots in several system tables and is useful for checking to see how full system tables have become if the system is under a heavy load.
- P*pid*   Restrict output to the process given by *pid*. Meaningful only when used with one of -p, -v, or -t.
- C*name*   Restrict output to those processes named *name*. Meaningful only when used with one of -p, -v, or -t.
- S*uid*   Display information pertaining to the share scheduler. If *uid* is present then output is restricted to processes with a share uid of *uid*. The output format includes the process name, process structure address (PROC), uid (UID), share group uid (SHGUID), scheduler cpu usage (CPU), and process id (PID). This flag is only useful if the system is running the share scheduler. This flag is generally used to track down processes that are incorrectly attached to the root inode. Processes running as the root inode will show up with UID == SHGUID == 0.

## FILES

<i>/vmunix</i>	kernel name list
<i>/dev/mem</i>	kernel data values
<i>/lib/kerneysyms/symdata_*</i>	kernel symbol addresses

## SEE ALSO

ps(1), stat(2), fstat(8), fs(5)  
 K. Thompson, "UNIX Implementation" in the *CONVEX UNIX Tutorial Papers*

## BUGS

It would be very useful if the system recorded "maximum occupancy" on the tables reported by -T; even more useful if these tables were dynamically allocated.

**NAME**

*putst* - manipulate kernel databases for striped disk partitions

**SYNOPSIS**

*putst* -vna [ /dev/rst? ... ]

**DESCRIPTION**

*putst* initializes a stripe description table in the UNIX kernel for a "striped disk" partition. Striped disk partitions are logical disk partitions which are interleaved over several physical disk partitions. Striped disk partitions are used to take advantage of performance improvements made possible by the parallel operation of the several disk arms which make up the striped disk partition. UNIX file systems can be mounted on striped disk partitions just as with "normal" disk partitions.

*putst* must be used before the striped partition can be initialized with *newfs*(8) or *mkfs*(8), or mounted as a UNIX file system, or otherwise accessed for reading and writing.

Normally *putst* is invoked at reboot time by the startup script */etc/rc* to load stripe tables from the stripe database file */etc/stripecap*. For each stripe device name */dev/rstX* given as an argument, *putst* searches for an entry named *stX* in */etc/stripecap*, and uses the parameters defined by that entry to initialize the striped file system information in the UNIX kernel.

If the *-a* switch is used, no file name arguments need be given, and *putst* automatically loads the stripe tables for */dev/rst0* through */dev/rst7* by using the entries *st0* through *st7* in */etc/stripecap* (if they exist).

If the *-v* switch is used, *putst* displays the information loaded into the stripe table.

If the *-n* switch is used, *putst* perform all of its normal actions, but will not actually update the kernel stripe table. This can be useful for checking the syntax of hand-generated *stripecap* entries.

For reasons of security, the stripe device entries ( */dev/rst?* ) are owned by the superuser; hence you must be the superuser to use *putst*.

Entries in */etc/stripecap* are created with the *newst*(8) utility.

**FILES**

*/etc/putst*  
*/etc/stripecap*

**SEE ALSO**

*getst*(8), *newst*(8), *st*(4), *stripecap*(5)

**NAME**

quot - summarize file system ownership

**SYNOPSIS**

`/usr/etc/quot [ -acfhnv ] [ filesystem ]`

**DESCRIPTION**

*quot* displays the number of blocks (1024 bytes) in the named *filesystem* currently owned by each user.

**OPTIONS**

- a Generate a report for all mounted file systems.
- c Display three columns giving file size in blocks, number of files of that size, and cumulative total of blocks in that size or smaller file.
- f Display count of number of files as well as space owned by each user.
- h Estimate the number of blocks in the file — this doesn't account for files with holes in them.
- n Run the pipeline `ncheck filesystem | sort +0n | quot -n filesystem` to produce a list of all files and their owners.
- v Display three columns containing the number of blocks not accessed in the last 30, 60, and 90 days.

**FILES**

<code>/etc/mtab</code>	mounted file systems
<code>/etc/passwd</code>	to get user names

**SEE ALSO**

ls(1), du(1), df(1), find(1)

**NAME**

quotacheck - check file system quota consistency

**SYNOPSIS**

```
/usr/etc/quotacheck [-v] [-p] filesystem...
/usr/etc/quotacheck [-v] [-p] -a
```

**DESCRIPTION**

*quotacheck* examines each file system, builds a table of current disk usage, and compares this table against that stored in the disk quota file for the file system. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated (the latter only occurs if an active file system is checked).

*quotacheck* expects each file system to be checked to have a quota file named *quotas* in the root directory. If none is present, *quotacheck* will ignore the file system.

*quotacheck* is normally run at boot time from the */etc/rc.local* file, see *rc(8)*, before enabling disk quotas with *quotaon(8)*.

*quotacheck* accesses the raw device in calculating the actual disk usage for each user. Thus, the file systems checked should be quiescent while *quotacheck* is running.

**OPTIONS**

- v Print a list of all file and block usage counts that are updated. The list consists of lines containing the disk partition name (if **-a** or **-p** was given), a user name, the old and new file count (if different), and the old and new block count (if different). File or block counts that have not changed will not be shown.
- p In this case *quotacheck* reads the table */etc/fstab* to determine which file systems to check. It uses the information there to inspect groups of disks in parallel taking maximum advantage of i/o overlap to check the file systems as quickly as possible.
- a check all the file systems indicated in */etc/fstab* to be read-write with disk quotas.

**FILES**

<i>quotas</i>	quota file at the file system root
<i>/etc/mstab</i>	mounted file systems
<i>/etc/fstab</i>	default file systems

**SEE ALSO**

quota(1), quotactl(2), quotaon(8), repquota(8), fstab(5)

**NAME**

quotaon, quotaoff - turn file system quotas on and off

**SYNOPSIS**

```
/usr/etc/quotaon [-v] filesystems . . .
/usr/etc/quotaon [-v] -a
/usr/etc/quotaoff [-v] filesystems . . .
/usr/etc/quotaoff [-v] -a
```

**DESCRIPTION OF QUOTAON**

*quotaon* announces to the system that disk quotas should be enabled on one or more file systems. The file systems specified must be mounted at the time. The file system quota files must be present in the root directory of the specified file system and be named *quotas*.

**OPTIONS TO QUOTAON**

-v display a message for each file system where quotas are turned on.  
 -a all file systems in */etc/fstab* marked read-write with quotas will have their quotas turned on. This is normally used at boot time to enable quotas.

**DESCRIPTION OF QUOTAOFF**

*quotaoff* announces to the system that file systems specified should have any disk quotas turned off.

**OPTIONS TO QUOTAOFF**

-v display a message for each file system affected.  
 -a force all file systems in */etc/fstab* to have their quotas disabled.

These commands update the status field of devices located in */etc/mtab* to indicate when quotas are on or off for each file system.

**FILES**

<i>quotas</i>	quota file at the file system root
<i>/etc/mtab</i>	mounted file systems
<i>/etc/fstab</i>	default file systems

**SEE ALSO**

quotactl(2), mtab(5), fstab(5)

**NAME**

*rc* – command script for auto-reboot and daemons

**SYNOPSIS**

*/etc/rc*  
*/etc/rc.local*

**DESCRIPTION**

*rc* is the command script which controls the automatic reboot and *rc.local* is the script holding commands which are pertinent only to a specific site.

When an automatic reboot is in progress, *rc* is invoked with the argument *autoboot* and runs a *fsck* with option *-p* to “preen” all the disks of minor inconsistencies resulting from the last system shutdown and to check for serious inconsistencies caused by hardware or software failure. If this auto-check and repair succeeds, then the second part of *rc* is run.

The second part of *rc*, which is run after a auto-reboot succeeds and also if *rc* is invoked when a single user shell terminates (see *init(8)*), starts all the daemons on the system, preserves editor files, configures the loopback device, and performs various other functions related to system startup. *rc.local* is executed immediately before any other commands after a successful *fsck*. Normally, the first command placed in the *rc.local* file defines the machine’s name, using *hostname(1)*.

**SEE ALSO**

*init(8)*, *reboot(8)*

**NAME**

*rdump* - dump file systems across the network

**SYNOPSIS**

*/etc/rdump* key [ *argument ...* ] filesystem

**DESCRIPTION**

*rdump* copies to magnetic tape all files changed in the *filesystem* after a certain date. *rdump* uses the same format and keys as *dump(8)*; in fact, *rdump* is simply an extension of *dump(8)*. To use *rdump*, you must use the *f* key with an argument of the form *machine:device*. If the *f* key is not specified, the default is *dumphost:/dev/rmt8*.

*rdump* creates a remote server, */etc/rmt*, on the client machine to access the tape device.

**FILES**

*~/rhosts*

This file exists on the server host with *rmt(8C)*. It contains a list of the Internet client host names having access to the server host.

*dumphost:/dev/rmt8*

The default remote tape drive to use if the *f* key is not specified.

**SEE ALSO**

*rhosts(5)*, *dump(8)*, *rmt(8C)*, *rrestore(8C)*

**DIAGNOSTICS**

Same as *dump(8)* with a few extra diagnostics related to the network.

**RESTRICTIONS**

The 100ips option of *dump(8)*, flag **I**, is not currently supported on remote devices.

**BUGS**

Since *rmt* was modified to perform write commands to tape in an asynchronous manner for performance improvements, error messages are not received by *rdump* until the subsequent write command is issued. This has a possible negative side effect if the very last write to tape fails. In this case, the error response will not be received until the close command is issued, and you will not receive the prompt to abort or try again. You will, however, see an error message indicating that something went wrong.

**NOTES**

Only the superuser may perform an *rdump*.

*rdump* is an optional product; for more information, contact your CONVEX sales representative.

**NAME**

reboot – UNIX bootstrapping procedures

**SYNOPSIS**

`/etc/reboot [-n] [-q]`

**DESCRIPTION**

UNIX is started by the Service Processor Unit (SPU). Since the system is not reenterable, it is necessary to read it in from disk each time it is to be bootstrapped.

**Rebooting a running system.** When a UNIX is running and a reboot is desired, `shutdown(8)` is normally used. If there are no users then `/etc/reboot` can be used. Reboot causes the disks to be synced, and then a multi-user reboot (as described below) is initiated. This causes a system to be booted and an automatic disk check to be performed. If all this succeeds without incident, the system is then brought up for many users.

Options to reboot are:

- `-n` option avoids the sync. It can be used if a disk or the processor is on fire.
- `-q` reboots quickly and ungracefully, without shutting down running processes first.

**Power fail and crash recovery.** Normally, the system will reboot itself at power-up or after crashes. Provided the auto-restart is enabled in the machine software front panel and the front panel key switch is in the "secure" position, an automatic consistency check of the file systems will be performed. Unless this fails, the system will resume multi-user operations.

In an emergency, the bootstrap methods described in the paper "Installing CONVEX UNIX" can be used to boot from a distribution tape. See the *CONVEX UNIX System Manager's Guide* for information on bootstrapping CONVEX UNIX.

**FILES**

`/vmunix` system code

**SEE ALSO**

`fsck(8)`, `init(8)`, `rc(8)`, `shutdown(8)`, `halt(8)`, `newfs(8)`

**NAME**

*remshent* - remove an entry from the shares data-base

**SYNOPSIS**

**remshent** [ **-p** *sharesfile* ] [ *name* | *uid* ] ...

**DESCRIPTION**

*remshent* removes entries from the shares data-base file */etc/shares*.

Arguments to *remshent* are user names or uids. If no arguments are given, *remshent* reads names from standard input, one name or uid per line.

If the **-p** option is specified, *remshent* will use *sharesfile* as the share data-base file.

When removing a user account, run *remshent* first, and then remove the user's entry in the password file, */etc/passwd*. If the entry in the password file is removed first, *remshent* will not be able to remove the user from the share data base.

**FILES**

<i>/etc/shares</i>	share information data base
<i>/etc/passwd</i>	password file

**SEE ALSO**

pl(1), lnode(5), share(5), shares(5), lim(8)

**NOTES**

*remshent* does not remove the user's entry from the shares file; instead, it zeroes all of the share data fields in the entry. The user is effectively removed from the share data base when the user's entry is removed from the password file.

*remshent* is part of the Share scheduler. Share is an optional product; please contact your CONVEX sales representative for more information.

**NAME**

*renice* - alter priority of running processes

**SYNOPSIS**

```
/etc/renice priority [ [ -p ] pid ... ] [ [ -g ] pgrp ... ] [ [ -u ] user ... ]
```

**DESCRIPTION**

*renice* alters the scheduling priority of one or more running processes. The *who* parameters are interpreted as process IDs, process group IDs, or user names. *renice*'ing a process group causes all processes in the process group to have their scheduling priority altered. *renice*'ing a user causes all processes owned by the user to have their scheduling priority altered. By default, the processes to be affected are specified by their process IDs. To force *who* parameters to be interpreted as process group IDs, a *-g* may be specified. To force the *who* parameters to be interpreted as user names, a *-u* may be given. Supplying *-p* will reset *who* interpretation to be (the default) process IDs. For example,

```
/etc/renice +1 987 -u daemon root -p 32
```

would change the priority of process IDs 987 and 32, and all processes owned by users *daemon* and *root*.

Users other than the superuser may only alter the priority of processes they own, and can only monotonically increase their "nice value" within the range 0 to PRIO\_MIN (64). (This prevents overriding administrative fiat.) The superuser may alter the priority of any process and set the priority to any value in the range PRIO\_MAX (-64) to PRIO\_MIN (64). Useful priorities are: 64 (the affected processes will run only when nothing else in the system wants to), 0 (the "base" scheduling priority), anything negative (to make things go very fast). However, when giving a task a very negative priority, other processes will receive a smaller portion of the cpu which may result in degradation of overall system performance.

**FILES**

/etc/passwd to map user names to user IDs

**SEE ALSO**

getpriority(2)

**BUGS**

If you make the priority very negative, then the process cannot be interrupted. To regain control you make the priority greater than zero. Non-superusers cannot increase scheduling priorities of their own processes, even if they were the ones that decreased the priorities in the first place.

**NAME**

repquota - summarize quotas for a file system

**SYNOPSIS**

```
/usr/etc/repquota [ -v ] filesystem...  
/usr/etc/repquota [ -v ] -a
```

**DESCRIPTION**

*repquota* prints a summary of the disc usage and quotas for the specified file systems. For each user the current number files and amount of space (in kilobytes) is printed, along with any quotas created with *edquota*(8).

**OPTIONS**

**-v** report all quotas, even if there is no usage.

Only the superuser may view quotas which are not their own.

**-a** report on all file systems indicated in */etc/fstab* to be read-write with quotas.

**FILES**

<i>quotas</i>	quota file at the file system root
<i>/etc/fstab</i>	default file systems

**SEE ALSO**

quota(1), quotactl(2), quotacheck(8), quotaon(8), edquota(8)

**NOTES**

Non-superusers should use "*quota -v*" for up-to-date calculations.

## NAME

restore – incremental file system restore

## SYNOPSIS

```
/etc/restore key [ name ... ]
```

## DESCRIPTION

*restore* reads tapes dumped with the *dump(8)* command. Its actions are controlled by the *key* argument. The *key* is a string of characters containing, at most one, function letter and possibly one or more function modifiers. Other arguments to the command are file or directory names specifying the files that are to be restored. Unless the *h* key is specified (see below), the appearance of a directory name refers to the files and recursively subdirectories of that directory.

The function portion of the key is specified by one of the following letters:

- r** The tape is read and loaded into the current directory. This should not be done lightly; the **r** key should only be used to restore a complete dump tape onto a clear file system or to restore an incremental dump tape after a full level zero restore. Thus:

```

/etc/newfs /dev/rda0g eagle
/etc/mount /dev/rda0g /mnt
cd /mnt
restore r

```

is a typical sequence to restore a complete dump. Another *restore* can be done to get an incremental dump in on top of this. Note that *restore* leaves a file *restoresymtab* in the root directory to pass information between incremental restore passes. This file should be removed when the last incremental tape has been restored. It should be obvious that the file system which is being restored from tape must not be larger than the disk file system into which it is being restored.

A *dump(8)*, followed by a *newfs(8)*, and a *restore* is used to change the size of a file system.

- R** *restore* requests a particular tape of a multi-volume set on which to restart a full restore (see the **r** key above). This allows *restore* to be interrupted and then restarted.
- x** The named files are extracted from the tape. If the named file matches a directory whose contents had been written onto the tape, and the **h** key is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored if possible. If no file argument is given, then the root directory is extracted, which results in the entire content of the tape being extracted, unless the **h** key has been specified.
- t** The names of the specified files are listed if they occur on the tape. If no file argument is given, then the root directory is listed, which results in the entire content of the tape being listed, unless the **h** key has been specified. Note that the **t** key replaces the function of the old *dumpdir* program.
- V** The dump tape(s) are verified for consistency. This key performs the same operation as the **x** key, except that nothing is written to disk. This option verifies that the tape can be read without error and the dump file format is correct. It does not verify the correctness of the directories and files contained in the dump file.
- i** This mode allows interactive restoration of files from a dump tape. After reading in the directory information from the tape, *restore* provides a shell-like interface that allows the user to move around the directory tree selecting files to be extracted. The available commands are given below; for those commands that require an argument, the default is the current directory.

**ls** [*arg*] – List the current or specified directory. Entries that are directories are appended with a */*. Entries that have been marked for extraction are prepended with a *\**. If

the verbose key is set the inode number of each entry is also listed.

**cd arg** – Change the current working directory to the specified argument.

**pwd** – Print the full pathname of the current working directory.

**add [arg]** – The current directory or specified argument is added to the list of files to be extracted. If a directory is specified, then it and all of its descendants are added to the extraction list (unless the **h** key is specified on the command line). Files that are on the extraction list are prepended with a **\***. when they are listed by **ls**.

**delete [arg]** – The current directory or specified argument is deleted from the list of files to be extracted. If a directory is specified, then it and all of its descendants are deleted from the extraction list (unless the **h** key is specified on the command line). The most expedient way to extract most of the files from a directory is to add the directory to the extraction list and then delete the files that are not needed.

**extract** – All of the files that are on the extraction list are extracted from the dump tape. *Restore* will ask which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume, and work towards the first volume.

**verbose** – The sense of the **v** key is toggled. When set, the verbose key causes the **ls** command to list the inode numbers of all entries. It also causes *restore* to print out information about each file as it is extracted.

**help** – List a summary of the available commands.

**quit** – *Restore* immediately exits, even if the extraction list is not empty.

The following characters may be used in addition to the letter that selects the function desired:

- v** Normally, *restore* does its work silently. The **v** (verbose) key causes *restore* to type the name of each file it treats preceded by its file type.
- c** *restore* converts old (4.1-BSD) directories to new (4.3+BSD) directory format.
- d** *restore* prints a great deal of debugging information on the controlling terminal.
- f** The next argument to *restore* is used as the name of the archive instead of */dev/rmt8*. If the name of the file is **-**, *restore* reads from standard input. Thus, *dump(8)* and *restore* can be used in a pipeline to dump and restore a file system with the command:
 

```
dump 0f - /usr | (cd /mnt; restore xf -)
```
- y** *restore* will not ask whether it should abort the restore if gets a tape error. It will always try to skip over the bad tape block(s) and continue as best it can.
- s n** *restore* skips to the *n*-th file on the tape before doing the restore. Tape files are numbered starting from **1** for the first file.
- m** *restore* will extract by inode numbers rather than by filename. This is useful if only a few files are being extracted, and you want to avoid regenerating the complete pathname to the file.
- h** *restore* extracts the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the tape.
- b** The next argument to *restore* is used as the blocking factor for tape records. This number indicates how many K-bytes are in a tape record. The default is 5.

- G** Defaults are set for a GCR (6250 bpi) *restore*. The tape unit is set to */dev/rmt16*, the density to 6250, and the tape blocking factor to 50.
- I** Puts the tape drive into 100 ips streaming mode. The default is 50 ips.

It should be stressed that the files that are extracted from a dumped tape will retain the entire path name with which they were dumped. For example, create a tape by *dumping* everything from a current working directory that contains the subdirectory *lib*, and its subdirectory *tmac*. Then perform the following steps :

```
% mkdir tmp tmp/tmac      # make directory to put files under
% cd tmp/tmac              # get there
% /etc/restore Gi          # interactive mode
restore> cd lib/tmac        # get to where the files are that we want
restore> add                # add all of them to the extraction list
restore> ls                 # list and verify that all files are starred
restore> extract            # get the starred files from the tape
restore> quit              # leave restore
```

This will extract the files on the tape from *lib/tmac/\** and place them under the current working directory, into *tmp/tmac/lib/tmac/\**. The files will **not** go directly under *tmp/tmac*, because the entire path saved on the tape is restored and appended to the *end* of the path of the current working directory.

## DIAGNOSTICS

Complaint about bad key characters.

Complaint if *restore* gets a read error. If *y* has been specified, or the user responds *y*, *restore* will attempt to continue the restore.

If the dump extends over more than one tape, *restore* will ask the user to change tapes. If the *x* or *i* key has been specified, *restore* will also ask which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume, and work towards the first volume.

There are numerous consistency checks that can be listed by *restore*. Most checks are self-explanatory or can "never happen". Common errors are:

### Converting to new file system format

A dump tape created from the old file system has been loaded. It is automatically converted to the new file system format.

<filename>: **not found on tape**

The specified filename was listed in the tape directory, but was not found on the tape. This is caused by tape-read errors while looking for the file, and from using a dump tape created on an active file system.

**expected next file** <inumber> got <inumber>

A file that was not listed in the directory showed up. This can occur when using a dump tape created on an active file system.

### Incremental tape too low

When doing incremental restore, a tape that was written before the previous incremental tape, or that has too low an incremental level, has been loaded.

### Incremental tape too high

When doing incremental restore, a tape that does not begin its coverage where the previous incremental tape left off, or that has too high an incremental level has been loaded.

**Tape read error while restoring** <filename>

**Tape read error while skipping over inode** <inumber>

**Tape read error while trying to resynchronize**

A tape read error has occurred. If a filename is specified, then its contents are probably partially wrong. If an inode is being skipped or the tape is trying to resynchronize, then no extracted files have been corrupted, though files may not be found on the tape.

**resync restore, skipped** <num> **blocks**

After a tape read error, *restore* may have to resynchronize itself. This message lists the number of blocks that were skipped over.

#### FILES

/dev/rmt8	the default tape drive
/tmp/rstidir*	file containing directories on the tape.
/tmp/rstmode*	owner, mode, and time stamps for directories.
./restoresymtab	information passed between incremental restores.

#### SEE ALSO

mtio(4), dump(8), mkfs(8), mount(8), newfs(8), newst(8)

#### BUGS

*restore* can get confused when doing incremental restores from dump tapes that were made on active file systems.

A level zero dump must be done after a full *restore*. Because *restore* runs in user code, it has no control over inode allocation; thus, a full *restore* must be done to get a new set of directories reflecting the new inode numbering, even though the contents of the files are unchanged.

If a non-standard blocking factor is used to dump a file system, it must also be used during a *restore*. *restore* should determine the blocking factor by itself.

## NAME

rexecd, in.rrexecd – remote execution server

## SYNOPSIS

/etc/rexecd

## DESCRIPTION

*rexecd* is the server for the *rexec(3X)* routine. The server provides remote execution facilities with authentication based on user names and encrypted passwords. It is started automatically as needed by *inetd(8C)* and initiates the following protocol:

- 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, unencrypted 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 initial socket 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 was password 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. This message is returned on the connection associated with the `stderr`, and is not preceded by a flag byte.

**SEE ALSO**

`rexec(3)`

**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 and password exchanges to be encrypted should be present.

**NAME**

*rivet* - Rivet a symbol table to the end of a *vmunix* file

**SYNOPSIS**

```
/bin/nm vmunix > nm.output  
/etc/rivet vmunix nm.output
```

**DESCRIPTION**

*rivet* is used to append a specially sorted symbol table to the end of a *vmunix* file. This symbol table is used by the SPU programs *sysboot*, *jpd*, and *jpstat* in order to allow fast symbol table lookups while minimizing the memory requirements on the SPU. Normally, *rivet* should only be invoked in the makefiles generated by *sysgen*.

**SEE ALSO**

*sysgen*(8)

## NAME

rlogind, in.rlogind – remote login server

## SYNOPSIS

`/etc/in.rlogind [ -ln ]`

## DESCRIPTION

*rlogind* is the server for the *rlogin*(1C) program. The server provides a remote login facility with authentication based on privileged port numbers from trusted hosts.

*rlogind* is invoked by *inetd*(8C) when a remote login connection is established, and initiates the following protocol:

- 1) The server checks the client's source port. If the port is not in the range 512-1023, the server aborts the connection.
- 2) The server checks the client's source address and requests the corresponding host name (see *gethostbyaddr*(3N), *hosts*(5) and *named*(8)). If the host name cannot be determined, the dot-notation representation of the host address is used.

Once the source port and address have been checked, *rlogind* allocates a pseudo terminal (see *pty*(4)), 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 *rshd*(8C), but if automatic authentication fails, it reprompts the user to login as one finds on a standard terminal line. The *-l* option prevents any authentication based on the user's ".rhosts" file, unless the user is logging in as the superuser.

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*(4) 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*(7). The screen or window size of the terminal is requested from the client, and window size changes from the client are propagated to the pseudo terminal.

Transport-level keepalive messages are enabled unless the *-n* option is present. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

## 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.

"Try again."

A *fork* by the server failed.

"/bin/sh: ..."

The user's login shell could not be started.

## SEE ALSO

*rlogin*(1C), *services*(5), *inetd*(8C), *ruserok*(3), *rshd*(8)

## 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.

A more extensible protocol should be used.

## NOTES

*rlogind* is an optional product; for more information, contact your CONVEX sales representative.

## NAME

rmt – remote magtape protocol module

## SYNOPSIS

/etc/rmt

## DESCRIPTION

*rmt* is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection. *rmt* is normally started up with an *rexec(3X)* or *rcmd(3X)* call.

The *rmt* program accepts requests specific to the manipulation of magnetic tapes, performs the commands, then responds with a status indication. All responses are in ASCII and in one of two forms. Successful commands have responses of

*A**number*\n

where *number* is an ASCII representation of a decimal number. Unsuccessful commands are responded to with

*E**error-number*\n*e**rror-message*\n,

where *error-number* is one of the possible error numbers described in *intro(2)* and *error-message* is the corresponding error string as printed from a call to *perror(3)*. The protocol is comprised of the following commands.

*O**device*\n*m**ode*\n

Open the specified *device* using the indicated *mode*. *device* is a full pathname and *mode* is an ASCII representation of a decimal number suitable for passing to *open(2)*. If a device had already been opened, it is closed before a new open is performed.

*C**device*\n Close the currently open device. The *device* specified is ignored.

*L**w**hence*\n*o**ffset*\n

Perform an *lseek(2)* operation using the specified parameters. The response value is that returned from the *lseek* call.

*W**count*\n

Write data onto the open device. *rmt* reads *count* bytes from the connection, aborting if a premature end-of-file is encountered. Write commands are different from the others, in that they are done in an asynchronous manner. That is, a response value is returned immediately after reading the bytes from the connection. If the previous write command was successful, then the number of bytes just read from the connection will be returned. If the previous write command failed, then an error response will be issued. The result of this asynchronous behavior is improved performance when performing remote dumps. A possible negative side effect of this system occurs if the final write to tape fails. This failure will not be reported until a close command is attempted. When running *rdump*, this error message will appear, but you will likely not be prompted to abort or continue, as you would if an earlier write command failed.

*R**count*\n

Read *count* bytes of data from the open device. If *count* exceeds the size of the data buffer (10 kilobytes), it is truncated to the data buffer size. *rmt* then performs the requested *read(2)* and responds with *A**count-read*\n if the read was successful; otherwise an error in the standard format is returned. If the read was successful, the data read is then sent.

*I**operation*\n*c**ount*\n

Perform a MTIOCOP *ioctl(2)* command using the specified parameters. The parameters are interpreted as the ASCII representations of the decimal values to place in the *mt\_op* and *mt\_count* fields of the structure used in the *ioctl* call.

The return value is the *count* parameter when the operation is successful.

**S** Return the status of the open device, as obtained with a `MTIOCGET ioctl` call. If the operation was successful, an "ack" is sent with the size of the status buffer, then the status buffer is sent (in binary).

Any other command causes *rmt* to exit.

#### DIAGNOSTICS

All responses are of the form described above.

#### SEE ALSO

`remd(3X)`, `rexec(3X)`, `mtio(4)`, `rdump(8C)`, `rrestore(8C)`

#### BUGS

People tempted to use this for a remote file access protocol are discouraged.

If the final write command to tape fails, the message on the *rdump* side will come after a close command has been issued, and you will not get prompted to abort or continue.

## NAME

route - manually manipulate the routing tables

## SYNOPSIS

*/etc/route* [ *-f* ] [ *-n* ] [ *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*(8C), should tend to this task.

*route* accepts two commands: *add*, to add a route, and *delete*, to delete a route.

All commands have the following syntax:

*/etc/route command* [ *net* | *host* ] *destination gateway* [ *metric* ]

where *destination* is the destination host or network, *gateway* is the next-hop gateway to which packets should be addressed, and *metric* is a count indicating the number of hops to the *destination*. The *metric* is required for *add* commands; it must be zero if the destination is on a directly-attached network, and nonzero if the route utilizes one or more gateways. If adding a route with *metric* 0, the gateway given is the address of this host on the common network, indicating the interface to be used for transmission. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. The optional keywords *net* and *host* force the destination to be interpreted as a network or a host, respectively. Otherwise, if the *destination* has a "local address part" of INADDR\_ANY, or if the *destination* is the symbolic name of a network, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host. If the route is to a destination connected via a gateway, the *metric* should be greater than 0. All symbolic names specified for a *destination* or *gateway* are looked up first as a host name using *gethostbyname*(3N). If this lookup fails, *getnetbyname*(3N) is then used to interpret the name as that of a network. If the destination of a gateway is specified as *default* that gateway is used if no other route can be found. This is not a good thing to do on large networks, because it can lead to packets being passed around unnecessarily.

*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.

The *-n* option prevents attempts to print host and network names symbolically when reporting actions.

## DIAGNOSTICS

"add [ *host* | *network* ] %s: gateway %s flags %x"

The specified route is being added to the tables. The values printed are from the routing table entry supplied in the *ioctl* call. If the gateway address used was not the primary address of the gateway (the first one returned by *gethostbyname*), the gateway address is printed numerically as well as symbolically.

"delete [ *host* | *network* ] %s: gateway %s flags %x"

As above, but when deleting an entry.

"%s %s done"

When the *-f* flag is specified, each routing table entry deleted is indicated with a message of this form.

"Network is unreachable"

An attempt to add a route failed because the gateway listed was not on a directly-connected network. The next-hop gateway must be given.

**“not in table”**

A delete operation was attempted for an entry which wasn't present in the tables.

**“routing table overflow”**

An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

**FILES**

<i>/vmunix</i>	Kernel name list
<i>/dev/mem</i>	Kernel data values
<i>/lib/kernsyms/symdata_*</i>	Kernel symbol addresses

**SEE ALSO**

intro(4N), routed(8C),

**NOTES**

*route* is an optional product; for more information, contact your CONVEX sales representative.

## NAME

routed - network routing daemon

## SYNOPSIS

```
/etc/routed [ -d ] [ -g ] [ -s ] [ -q ] [ -t ] [ logfile ]
```

## DESCRIPTION

*routed* is invoked at boot time to manage the network routing tables. The routing daemon uses a variant of the Xerox NS Routing Information Protocol in maintaining up to date kernel routing table entries. It uses a generalized protocol capable of use with multiple address types, but is currently used only for Internet routing within a cluster of networks.

In normal operation *routed* listens on the *udp(4p)* socket for the *route* service (see *services(5)*) for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When *routed* is started, it uses the *SIOCGIFCONF ioctl* to find those directly connected interfaces configured into the system and marked "up" (the software loopback interface is ignored). If multiple interfaces are present, it is assumed that the host will forward packets between networks. *routed* then transmits a *request* packet on each interface (using a broadcast packet if the interface supports it) and enters a loop, listening for *request* and *response* packets from other hosts.

When a *request* packet is received, *routed* formulates a reply based on the information maintained in its internal tables. The *response* packet generated contains a list of known routes, each marked with a "hop count" metric (a count of 16, or greater, is considered "infinite"). The metric associated with each route returned provides a metric *relative to the sender*.

*Response* packets received by *routed* are used to update the routing tables if one of the following conditions is satisfied:

- (1) No routing table entry exists for the destination network or host, and the metric indicates the destination is "reachable" (i.e. the hop count is not infinite).
- (2) The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.
- (3) The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.
- (4) The new route describes a shorter route to the destination than the one currently stored in the routing tables; the metric of the new route is compared against the one stored in the table to decide this.

When an update is applied, *routed* records the change in its internal tables and updates the kernel routing table. The change is reflected in the next *response* packet sent.

In addition to processing incoming packets, *routed* also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated throughout the local internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks. The response is sent to the broadcast address on nets capable of that function, to the destination address on point-to-point links, and to the router's own address on other networks. The normal routing tables are bypassed when sending gratuitous responses. The reception of responses on each network is used to determine that the network and interface are functioning correctly. If no response is received on an interface, another route may be chosen to route around the interface, or the route may be dropped if no alternative is available.

*routed* supports several options:

- d Enable additional debugging information to be logged, such as bad packets received.
- g This flag is used on internetwork routers to offer a route to the "default" destination. This is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers.
- s Supplying this option forces *routed* to supply routing information whether it is acting as an internetwork router or not. This is the default if multiple network interfaces are present, or if a point-to-point link is in use.
- q This is the opposite of the -s option.
- t If the -t option is specified, all packets sent or received are printed on the standard output. In addition, *routed* will not divorce itself from the controlling terminal so that interrupts from the keyboard will kill the process.

Any other argument supplied is interpreted as the name of file in which *routed*'s actions should be logged. This log contains information about any changes to the routing tables and, if not tracing all packets, a history of recent messages sent and received which are related to the changed route.

In addition to the facilities described above, *routed* supports the notion of "distant" *passive* and *active* gateways. When *routed* is started up, it reads the file */etc/gateways* to find gateways which may not be located using only information from the SIOGIFCONF *ioctl*. Gateways specified in this manner should be marked *passive* if they are not expected to exchange routing information, while gateways marked *active* should be willing to exchange routing information (i.e. they should have a *routed* process running on the machine). Routes through *passive* gateways are installed in the kernel's routing tables once upon startup. Such routes are not 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. Gateways marked *external* are also *passive*, but are not placed in the kernel routing table nor are they included in routing updates. The function of *external* entries is to inform *routed* that another routing process will install such a route, and that alternate routes to that destination should not be installed. Such entries are only required when both routers may learn of routes to the same destination.

Internetwork routers that are directly attached to the Arpanet or Milnet should use the Exterior Gateway Protocol (EGP) to gather routing information rather than using a static routing table of *passive* gateways. EGP is required in order to provide routes for local networks to the rest of the Internet system. Sites needing assistance with such configurations should contact the CONVEX Technical Assistance Center.

#### FILES

*/etc/gateways* for distant gateways

#### SEE ALSO

"Internet Transport Protocols", XSIIS 028112, Xerox System Integration Standard.  
 udp(4), icmp(4), htable(8), gateways(5)

#### BUGS

The kernel's routing tables may not correspond to those of *routed* when redirects change or add routes. *routed* should note any redirects received by reading the ICMP packets received via a raw socket.

*routed* should incorporate other routing protocols, such as EGP. Using separate processes for each requires configuration options to avoid redundant or competing routes.

*routed* should listen to intelligent interfaces, such as an IMP, to gather more information. It does not always detect unidirectional failures in network interfaces (e.g., when the output side fails).

NOTES

*routed* is an optional product; for more information, contact your CONVEX sales representative.

**NAME**

rrestore - restore a file system dump across the network

**SYNOPSIS**

`/etc/rrestore` key [ name ... ]

**DESCRIPTION**

*rrestore* obtains from magnetic tape files saved by a previous *dump*(8) or *rdump*(8c). *rrestore* uses the same format and keys as *restore*(8); in fact, *rrestore* is simply an extension of *restore*(8). To use *rrestore*, you must use the *f* key with an argument of the form *machine:device*. If the *f* key is not specified, *dumphost:/dev/rmt8* is used.

*rrestore* creates a remote server, */etc/rmt*, on the client machine to access the tape device.

**FILES**

`~/rhosts`

This file exists with *rmt*(8C) on the server host. It contains a list of the Internet client host names having access to the server host.

`dumphost:/dev/rmt8`

The default remote tape drive to use if the *f* key is not specified.

**SEE ALSO**

*rhosts*(5), *restore*(8), *rmt*(8C)

**DIAGNOSTICS**

Same as *restore*(8) with a few extra diagnostics related to the network.

**RESTRICTIONS**

The 100ips option of *restore*(8), flag **I**, is not currently supported on remote devices.

**NOTES**

Only the superuser may perform an *rrestore*.

*rrestore* is an optional product; for more information, contact your CONVEX sales representative.

## NAME

rshd, in.rshd - remote shell server

## SYNOPSIS

/etc/in.rshd [-d] host.port

## DESCRIPTION

*rshd* is the server for the *rcmd(3X)* routine and, consequently, for the *rsh(1C)* program. The server provides remote execution facilities with authentication based on privileged port numbers.

*-l* prevents any authentication based on the user's ".rhosts" file, unless the user is logging in as the superuser.

*-n* disables transport-level keepalive messages. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

*rshd* is invoked by *inetd(8C)* when a remote shell connection is established, and initiates the following protocol:

- 1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection. The clients address and port number are passed as arguments to *rshd* by *inetd* in the form "host.port" with host in hex and port in decimal.
- 2) The server reads characters from the socket up to a null (" ") byte. The resultant string is interpreted as an ASCII number, base 10.
- 3) If the number received in step 2 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 and requests the corresponding host name (see *gethostbyaddr(3N)*, *hosts(5)* and *named(8)*). If the hostname cannot be determined, the dot-notation representation of the host address is used.
- 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 client'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 server'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) *rshd* 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 superuser, (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 superuser, 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 *rshd*.

## DIAGNOSTICS

Except for the last one listed below, 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).

**“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.

**“<shellname>: ...”**

The user's login shell could not be started. This message is returned on the connection associated with the *stderr*, and is not preceded by a flag byte.

**“/bin/sh: ...”**

The user's login shell could not be started.

**SEE ALSO**

rsh(1C), rcmd(3X), services(5), inetd(8C)

**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.

**NOTES**

*rshd* is an optional product; for more information, contact your CONVEX sales representative.

## NAME

*rwhod* – system status server

## SYNOPSIS

*/etc/rwhod*

## DESCRIPTION

*rwhod* is the server which maintains the database used by the *rwho*(1C) and *ruptime*(1C) 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 server transmits and receives messages at the port indicated in the “*rwho*” service specification; see *services*(5). 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 *w*(1) program, and represent load averages over the 5, 10, and 15 minute intervals prior to a server's transmission; they are multiplied by 100 for representation in an integer. The host name included is that returned by the *gethostname*(2) system call, with any trailing domain name omitted. 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*(5) entry for each non-idle terminal line and a value indicating the time in seconds since a character was last received on the terminal line.

Messages received by the *rwho* server are discarded unless they originated at an *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 3 minutes. *rwhod* performs an *nlist*(3) on */vmunix* every 30 minutes to guard against the possibility that this file is not the system image currently operating.

**FILES**

<i>/vmunix</i>	Kernel name list
<i>/dev/mem</i>	Kernel data values
<i>/lib/kernsyms/symdata_*</i>	Kernel symbol addresses

**SEE ALSO**

*rwho(1C)*, *ruptime(1C)*

**BUGS**

There should be a way to relay status information between networks. Status information should be sent only upon request rather than continuously. People often interpret the server dying or network communication failures as a machine going down.

**NOTES**

*rwhod* is an optional product; for more information, contact your CONVEX sales representative.

## NAME

sa, accton - system accounting

## SYNOPSIS

```
/etc/sa [ -abcdDefFgijkKlmnrstuv ] [ acctfile ]
/etc/accton [ acctfile ]
```

## DESCRIPTION

With an argument naming an existing *file*, *accton* causes system accounting information for every process executed to be placed at the end of the file. If no argument is given, accounting is turned off.

*sa* reports on, cleans up, and generally maintains accounting files.

*sa* is able to condense the information in the accounting file into 2 summary files. The accounting file is assumed to be */usr/adm/acct* unless an alternate accounting file is specified in the last argument.

The accounting system automatically stops placing data at the end of */usr/adm/acct* if the disk space in the */usr/adm* directory exceeds 98% capacity. When this occurs, accounting will be automatically restarted when the available space drops back to 96% of capacity. To avoid losing accounting data, allow plenty of extra disk space in the */usr/adm* directory.

*/usr/adm/savacct* contains a count of the number of times each command was executed, CPU time consumed, total I/O, and memory usage. The other summary file is */usr/adm/usracct*, which contains similar information for each user and also for each unique group-activity combination. The condensation of the accounting file into summary files is desirable because on a large system the accounting file can grow by 100 blocks per day. The summary files are normally read before the accounting file, so the reports include all available information.

Output fields are labeled: "CPU" for the sum of user+system time (in minutes), "re" for real time (also in minutes), "k" for CPU-time averaged core usage (in 1-Kbyte units), "avio" for average number of I/O operations per execution. Various command-line switches can cause fields to appear labeled "tio" for total I/O operations, "k\*sec" for CPU storage integral (kilo-core seconds), and "u" and "s" for user and system CPU time alone (both in minutes). See *acct(5)* for more information on the CPU storage intergral.

*sa* recognizes the following command-line switches:

- a Place all command names containing unprintable characters and those used only once under the name "\*\*\*other".
- b Sort output by sum of user and system time divided by number of calls. Default sort is by sum of user and system times.
- c Besides total user, system, and real time for each command print percentage of total time over all commands.
- d Sort by average number of disk I/O operations.
- D Print and sort by total number of disk I/O operations.
- e Echo records exactly as they appear in the accounting file; this option may be used to convert the accounting file to ASCII text. The fields are, in order: command name, timestamp when execution started, total CPU time in seconds, TTY name, user ID, group ID, activity ID, elapsed time in seconds, CPU storage integral (kilo-core seconds), disk I/O blocks, system CPU time in seconds, the average level of concurrency (a floating point value between one and the number of processors in the system, indicating the average number of processors dedicated to the process at each clock interval; if this value is zero, then that particular accounting record was collected prior to CONVEX UNIX Release 7.0), and a character flag indicating in which scheduling mode the command was

executed (**f** signifies that it ran as a fixed-schedule job at least part of the time; **d** represents a dynamic-schedule job (dynamic is the default scheduling mode)). User CPU time may be computed by subtracting system CPU time from total CPU time. See *acct(5)* for more information on the accounting file.

- f** Force no interactive threshold compression with **-v** flag.
- Fc** Append the character *c* to the summary files, thus summarizing data in files */usr/adm/savacctc* and */usr/adm/usracctc*.
- g** For each unique group-activity combination, print a record containing the group name, activity name, number of processes, CPU minutes, total I/O, and memory usage.
- i** Don't read in summary files.
- j** Instead of total minutes time for each category, give seconds per call.
- k** Sort by CPU-time average memory usage.
- K** Print and sort by CPU-storage integral.
- l** Separate system and user time; normally they are combined.
- m** Print number of processes, CPU minutes, total I/O, and memory usage for each user.
- n** Sort by number of calls.
- r** Reverse order of sort.
- s** Use the data in the accounting file to create a summary of process data in file */usr/adm/savacct*, and a summary of user and group-activity data in file */usr/adm/usracct*. After the summaries have been created, remove all data in the accounting file.
- t** For each command report ratio of real time to the sum of user and system times.
- u** Superseding all other flags except **-e**, print the user ID and command name for each command in the accounting file.
- v** Followed by a number *n*, type the name of each command used *n* times or fewer. Await a reply from the terminal; if it begins with "y", add the command to the category **"\*\*junk\*\*"**. This is used to strip out garbage.

#### FILES

<i>/usr/adm/acct</i>	raw accounting
<i>/usr/adm/savacct</i>	per process summary
<i>/usr/adm/usracct</i>	per user and per group-activity summary

#### SEE ALSO

*lastcomm(1)*, *mpa(1)*, *acct(2)*, *acct(5)*, *activities(5)*, *ac(8)*,  
 "Accounting Reports" chapter in *Managing ConvexOS: Operations Guide*.

## NAME

*stat*, *seestat* - gather system statistics and print graphs of system statistics

## SYNOPSIS

```
seestat [ options ]
stat [ -i interval ]
```

## DESCRIPTION

*seestat* uses data collected by *stat* to print graphs of system use and performance. *stat* collects system data every *interval* seconds. If no *interval* is specified the default is thirty seconds. Currently *seestat* only supports output to a Printronix printer. It opens a pipe directly to *lpr -l*, so the environment variable **PRINTER** must be set to the name of a Printronix printer. The options of *seestat* are:

- anumb* Specify a number which varies the smoothness of the curve through a moving average algorithm. A value of 1 will print the graph with no smoothing. The default value is 12.
- ddate* Specify the date to print a graph for, in the form **month/day[/year]**. The default is yesterday.
- mmax* Specify the maximum y-axis position. The defaults are dependent on the statistic chosen. The argument **mm** causes the highest data point to be taken as the top of the axis.
- nnumb* Specify the number of the daily graph to be printed:

- |                                |                          |
|--------------------------------|--------------------------|
| 1. Buffered I/O latency        | 2. Buffered I/O Mb/s     |
| 3. Buffered I/O hits %         | 4. Disk Mb/s             |
| 5. Stripe Mb/s                 | 6. Network packets in    |
| 7. Network packets out         | 8. Network input errors  |
| 9. Network output errors       | 10. Network collisions   |
| 11. CPU user time pct          | 12. CPU niced time pct   |
| 13. CPU system time pct        | 14. CPU idle time pct    |
| 15. Virtual memory activated   | 16. Free virtual memory  |
| 17. Page ins/sec               | 18. Page outs/sec        |
| 19. Device interrupts/sec      | 20. System calls/sec     |
| 21. Load average               | 22. Disk request size    |
| 23. Real memory available      | 24. System users         |
| 25. Page reclaims/sec          | 26. Page faults/sec      |
| 27. Page hits %                | 28. Total # processes    |
| 29. Pages scanned by clock/sec | 30. Context switches/sec |

- tdots* Specify the size of the time- (x-) axis in dots. Defaults to 600.
- wnumb* Specify weekly graph to print. Defaults to 1.
- ydots* Specify the size of the y-axis in dots. Defaults to 190.

The ordering of the arguments to *seestat* is significant. The **n** and **w** options will actually cause the printing of graphs as the options are encountered in the command line. Therefore, each graph is drawn with only the options activated that *precede* the **n** or **w** option. Later options will affect only graphs printed with a later **n** or **w** option.

## EXAMPLES

To print a graph of yesterday's load average with a moving average of 20, enter the following command:

```
seestat -a20 -n21
```

The similar command

```
seestat -n21 -a20
```

will *not* give the same results, because the **a** option *follows* the **n**. The **a** in this case has no effect. For a weekly summary graph beginning June 21st of the current year showing the number of system users, type:

```
seestat -d6/21 -w24
```

It is also possible to print several graphs in one command. For example,

```
seestat -a21 -d3/11 -w21 -d3/18 -w21 -d3/12 -n24
```

will print three graphs, two weekly summaries of load average for the weeks beginning March 11th and March 18th of the current year, and a graph of daily system users for March 12th.

#### FILES

/dev/kmem, /vmunix, /etc/utmp	data sources
/usr/adm/stat/stats*	data stored by <i>stat</i>
/lib/kernsyms/symdata_*	Kernel symbol addresses

#### SEE ALSO

vmstat(1), uptime(1)

## NAME

sendmail - send mail over the internet

## SYNOPSIS

```
/usr/lib/sendmail [ flags ] [ address ... ]
mailq [-v]
```

## DESCRIPTION

*sendmail* sends a message to one or more *recipients*, routing the message over whatever networks are necessary. *sendmail* does internetwork forwarding as necessary to deliver the message to the correct place.

*sendmail* is not intended as a user interface routine; other programs provide user-friendly front ends; *sendmail* is used only to deliver pre-formatted messages.

With no flags, *sendmail* reads its standard input up to an end-of-file or a line consisting only of a single dot and sends a copy of the message found there to all of the addresses listed. It determines the network(s) to use based on the syntax and contents of the addresses.

Local addresses are looked up in a file and aliased appropriately. Aliasing can be prevented by preceding the address with a backslash. Normally the sender is not included in any alias expansions, e.g., if 'john' sends to 'group', and 'group' includes 'john' in the expansion, then the letter will not be delivered to 'john'.

Flags are:

- ba           Go into ARPANET mode. All input lines must end with a CR-LF, and all messages will be generated with a CR-LF at the end. Also, the "From:" and "Sender:" fields are examined for the name of the sender.
- bd           Run as a daemon. This requires Berkeley IPC. *sendmail* will fork and run in background listening on socket 25 for incoming SMTP connections. This is normally run from */etc/rc*. When run in this mode, *sendmail* alters its "name" (as seen by *ps(1)*) to show what task it is performing. The most common names are:
  - accepting connections
  - rejecting connections: load average <average>
  - running queue
  - svrsmtp <hostname>
- bi           Initialize the alias database.
- bm           Deliver mail in the usual way (default).
- bp           Print a listing of the queue.
- bs           Use the SMTP protocol as described in RFC821 on standard input and output. This flag implies all the operations of the -ba flag that are compatible with SMTP.
- bt           Run in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.
- bv           Verify names only - do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists on the local machine.
- bz           Create the configuration freeze file.
- Cfile       Use alternate configuration file. *sendmail* refuses to run as root if an alternate configuration file is specified. The frozen configuration file is bypassed.
- dX           Set debugging value to X.

- F***fullname*      Set the full name of the sender.
- f***name*          Sets the name of the "from" person (i.e., the sender of the mail). **-f** can only be used by "trusted" users (normally *root*, *daemon*, and *network*) or if the person you are trying to become is the same as the person you are.
- h***N*              Set the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop. If not specified, "Received:" lines in the message are counted.
- n**                Don't do aliasing.
- o***x value*        Set option *x* to the specified *value*. Options are described below.
- q**[*time*]         Processed saved messages in the queue at given intervals. If *time* is omitted, process the queue once. *Time* is given as a tagged number, with 's' being seconds, 'm' being minutes, 'h' being hours, 'd' being days, and 'w' being weeks. For example, "-q1h30m" or "-q90m" would both set the timeout to one hour thirty minutes. If *time* is specified, *sendmail* will run in background. This option can be used safely with **-bd**.
- r***name*          An alternate and obsolete form of the **-f** flag.
- t**                Read message for recipients. To:, Cc:, and Bcc: lines will be scanned for recipient addresses. The Bcc: line will be deleted before transmission. Any addresses in the argument list will be suppressed, that is, they will *not* receive copies even if listed in the message header.
- v**                Go into verbose mode. Alias expansions will be announced, etc.

There are also a number of processing options that may be set. Normally these will only be used by a system administrator. Options may be set either on the command line using the **-o** flag or in the configuration file. These are described in detail in the *Sendmail Installation and Operation Guide*. The options are:

- A***file*            Use alternate alias file.
- c**                On mailers that are considered "expensive" to connect to, don't initiate immediate connection. This requires queuing.
- d***x*              Set the delivery mode to *x*. Delivery modes are 'i' for interactive (synchronous) delivery, 'b' for background (asynchronous) delivery, and 'q' for queue only - i.e., actual delivery is done the next time the queue is run.
- D**                Try to automatically rebuild the alias database if necessary.
- e***x*                Set error processing to mode *x*. Valid modes are 'm' to mail back the error message, 'w' to "write" back the error message (or mail it back if the sender is not logged in), 'p' to print the errors on the terminal (default), 'q' to throw away error messages (only exit status is returned), and 'e' to do special processing for the BerkNet. If the text of the message is not mailed back by modes 'm' or 'w' and if the sender is local to this machine, a copy of the message is appended to the file "dead.letter" in the sender's home directory.
- F***mode*            The mode to use when creating temporary files.
- f**                Save UNIX-style From lines at the front of messages.
- g***N*              The default group id to use when calling mailers.
- H***file*            The SMTP help file.
- i**                Do not take dots on a line by themselves as a message terminator.

<b>L</b> <i>n</i>	The log level.
<b>m</b>	Send to "me" (the sender) also if I am in an alias expansion.
<b>o</b>	If set, this message may have old style headers. If not set, this message is guaranteed to have new style headers (i.e., commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.
<b>Q</b> <i>queuedir</i>	Select the directory in which to queue messages.
<b>r</b> <i>timeout</i>	The timeout on reads; if none is set, <i>sendmail</i> will wait forever for a mailer. This option violates the word (if not the intent) of the SMTP specification, show the timeout should probably be fairly large.
<b>S</b> <i>file</i>	Save statistics in the named file.
<b>s</b>	Always instantiate the queue file, even under circumstances where it is not strictly necessary. This provides safety against system crashes during delivery.
<b>T</b> <i>time</i>	Set the timeout on undelivered messages in the queue to the specified time. After delivery has failed (e.g., because of a host being down) for this amount of time, failed messages will be returned to the sender. The default is three days.
<b>t</b> <i>stz, dtz</i>	Set the name of the time zone.
<b>u</b> <i>N</i>	Set the default user id for mailers.

In aliases, the first character of a name may be a vertical bar to cause interpretation of the rest of the name as a command to pipe the mail to. It may be necessary to quote the name to keep *sendmail* from suppressing the blanks from between arguments. For example, a common alias is:

```
msgs: "|/usr/ucb/msgs -s"
```

Aliases may also have the syntax "*:include:filename*" to ask *sendmail* to read the named file for a list of recipients. For example, an alias such as:

```
poets: ":include:/usr/local/lib/poets.list"
```

would read */usr/local/lib/poets.list* for the list of addresses making up the group.

*sendmail* returns an exit status describing what it did. The codes are defined in *<sysexit.h>*

<b>EX_OK</b>	Successful completion on all addresses.
<b>EX_NOUSER</b>	User name not recognized.
<b>EX_UNAVAILABLE</b>	Catchall meaning necessary resources were not available.
<b>EX_SYNTAX</b>	Syntax error in address.
<b>EX_SOFTWARE</b>	Internal software error, including bad arguments.
<b>EX_OSERR</b>	Temporary operating system error, such as "cannot fork".
<b>EX_NOHOST</b>	Host name not recognized.
<b>EX_TEMPFAIL</b>	Message could not be sent immediately, but was queued.

If invoked as *mailq*, *sendmail* will print the contents of the mail queue.

## FILES

Except for */usr/lib/sendmail.cf*, these pathnames are all specified in */usr/lib/sendmail.cf*. Thus, these values are only approximations.

<i>/usr/lib/aliases</i>	raw data for alias names
<i>/usr/lib/aliases.pag</i>	
<i>/usr/lib/aliases.dir</i>	data base of alias names
<i>/usr/lib/sendmail.cf</i>	configuration file

/usr/lib/sendmail.fc	frozen configuration
/usr/lib/sendmail.hf	help file
/usr/lib/sendmail.st	collected statistics
/usr/spool/mqueue/*	temp files

**SEE ALSO**

binmail(1), newaliases(1), mail(1), rmail(1), syslog(3), aliases(5), mailaddr(7), rc(8);

**NAME**

*sethbb* - Set the number of HASP Host communications boards

**SYNOPSIS**

*sethbb number\_of\_boards*

**DESCRIPTION**

*sethbb* is a *csk(1)* script that sets the number of communications boards in the *current\_rmts(5)* file.

*number\_of\_boards* is the number of boards in a Convex computer. *sethbb* will fail if this parameter is omitted or is non-numeric.

**FILES**

<i>/usr/hasp/bin/sethbb</i>	This script
<i>/usr/hasp/etc/current_rmts</i>	HASP Host board database

**SEE ALSO**

*current\_rmts(5)*, *hhstart(8)*, *hhstop(8)*

**NOTES**

*sethbb* is part of the CONVEX HASP Host Communications Product. HASP Host is an optional product; for more information, contact your CONVEX sales representative.

## NAME

sharecf, sharesh - configure and display a Share configuration

## SYNOPSIS

**sharecf** [-n] [-v] [-p *sharesfile*] [-f *inputfile*] [-r *rootgroup*] [-s *shares*]

**sharesh** [-c] [-p *sharesfile*] [**lname=name**|**uid=uid**|**gname=groupname**|**gid=gid**]...

## DESCRIPTION

*sharecf* is a tool to help setup and maintain a CONVEX Share Scheduler system. *sharecf* uses a special-format file to describe a Share configuration; the configuration file syntax is C-like and is very flexible.

Entries in the configuration file describe Share scheduling groups and users.

A Share scheduling group entry has the form

```
user_identifier [sgroup=sharegroup] [shares=n] [mshares=m] [notshared]
[{member[member]... }];
```

A user entry has the form

```
user_identifier [shares=n] [sgroup=sharegroup] [notshared];
```

An entry must end in a ; to separate it from other entries.

*user\_identifier* is the name of the user or scheduling group and must be one of the following: **lname=name**, **uid=uid**, **gname=groupname** or **gid=gid**. *name* must be a valid login name (as defined in the password file */etc/passwd*.) *uid* must be a valid user id. *groupname* must be a valid ConvexOS group name (as defined in the groups file */etc/groups*.) *gid* must be a valid group id. As a convenience, users may also be referred to simply by their login name or uid.

If **gname** or **gid** is specified, *groupname* or *gid* is expanded in place to the list of members in the group.

**sgroup** specifies the parent scheduling group of the entry. **shares** specifies that the user or scheduling group will receive *n* shares. **mshares** specifies that by default, each *member* of the scheduling group will receive *m* shares apiece (see the discussion of the { } option below.) *n* and *m* must be integers.

The **notshared** option, if given, defines that the share scheduling group is NOTSHARED. By default, scheduling groups are not NOTSHARED. Be careful not to make a user entry NOTSHARED - that user will receive 0% of the cpu.

A definition of a Share scheduling group may also include a list of members for that group. A list of members is started with { and ends with }; a *member* may be another scheduling group or a list of users or a combination of both. By default, a *member* receives *m* shares (as defined by **mshares**) and is a part of the enclosing scheduling group definition. The value of **mshares** for individual entries can be overridden with the **shares** specifier. You can also override the default scheduling group with the **sgroup** specifier.

Where no specifiers are given, the defined defaults are used instead. By default root is the parent scheduling group of all entries unless overridden; entries receive 100 shares by default. If the **-s** switch is given, the default number of shares will be set to *n*. If the **-r** switch is given, the default scheduling group will be set to *groupname*.

*sharecf* reads stdin by default; an alternate configuration file may be specified using the **-f** switch.

*sharecf* uses *lim(8)* to change Share information. If the **-v** switch is given, *sharecf* will echo the commands as they are processed. If the **-n** switch is used, *sharecf* will not run any commands but will print them to standard output. This can aid in verifying that the configuration file is correct.

*sharesh* displays the current Share configuration. Output from *sharesh* has the same format as input files of *sharecf*. If the **-c** switch is given, *sharesh* will print a list of *lim(8)* commands that would produce the current configuration.

By default, *sharesh* displays information about all users and scheduling groups. Subsets of the current configuration can be selected using the **lname**, **uid**, **gid** or **gname** options. These options can be repeated and only the requested portions of the scheduling tree will be displayed.

Both commands accept the **-p** switch. If specified *sharecf* or *sharesh* will use *sharesfile* as the share data base file.

#### EXAMPLES

The following configuration file will give *jones* 150 shares and put him in the scheduling group *primaryg*. Uid 473 and the user named *mikew* will receive 100 shares and will be part of the *root* scheduling group (the defaults apply since no **shares** or **sgroup** specifier was given.) On line 3, the scheduling group *primaryg* will have 100 shares and will be a member of the root scheduling group; its members will receive 175 shares by default. The user named *applet* will be a member of the *primaryg* scheduling group as will each member of the ConvexOS group *lp*. Each of those users will receive 175 shares apiece. On line 4, a new group called *secondg* is defined; it will receive 175 shares (the default) and its members will receive 50 shares. *stever* will be a member of *secondg* and will receive 50 shares. Skipping a few lines, *andybob* and uid *356* are members of the scheduling group *primaryg* and receives 175 shares apiece. And so on.

```
lname=jones shares=150 sgroup=primaryg;
uid=473; lname=mikew;
lname=primaryg shares=100 mshares=175 sgroup=root {
    applet; gname=lp;
    lname=secondg mshares=50 {
        stever;
        lname=thirdg mshares=100 {
            toni;
        };
    };
    andybob; 356;
};
```

To display the current Share configuration:

```
sharesh
```

To display the current configuration for the group *lp*:

```
sharesh gname=lp
```

#### FILES

<i>/etc/shares</i>	share database file.
<i>/etc/passwd</i>	for user names and IDs.
<i>/etc/group</i>	for group names and group IDs.

#### SEE ALSO

*charges(1)*, *pl(1)*, *pwintf(1)*, *lnode(5)*, *shares(5)*, *lim(8)*

#### NOTES

*sharecf* and *sharesh* are part of the Share scheduler. Share is an optional product; please contact

your CONVEX sales representative for more information.

**NAME**

sharer - clean up defunct user *lnode* structures

**SYNOPSIS**

**sharer**

**DESCRIPTION**

*sharer* is invoked at system start-up and runs as a daemon in the background, cleaning up kernel *lnode* structures after users have logged off. *Lnodes* are installed for users by *login(1)*.

On startup, *sharer* installs a default user *lnode* (called "Other") for use by *login* when more than the allocated number of users are attempting to log in. *sharer* is attached to this *lnode*, so it attempts to keep its effective share between some bounds by adjusting the *lnode's* usage. Thus users who get allocated to this *lnode* will be guaranteed some minimum share of the resources.

*sharer* also maintains a *checkpoint* file of active users, so that after a system crash charges accumulated by users active at the time of the crash are not lost. This file, if present, is read when *sharer* is first invoked, and active *lnodes* found in it are updated in the shares data-base.

*sharer* scans for dead user *lnodes* updating them in the share data base file as they appear, and checkpointing active *lnodes* every few minutes.

*sharer* catches the system termination signal, which causes it to remove the checkpoint file, update the details for any remaining active *lnodes*, and terminate.

**FILES**

<i>/etc/shares</i>	share data base file.
<i>/etc/lnodes.chkpt</i>	checkpoint file for active <i>lnodes</i> .

**SEE ALSO**

*login(1)*, *pl(1)*, *limits(2)*, *setupshares(3)*, *lnode(5)*, *share(5)*, *shares(5)*, *lim(8)*, *remshent(8)*

**BUGS**

If *sharer* should die for any reason, then new users logging into the system will gradually consume all available kernel *lnodes*.

**NOTES**

*sharer* is part of the Share scheduler. Share is an optional product; please contact your CONVEX sales representative for more information.

## NAME

shutdown - close down the system at a given time

## SYNOPSIS

`/etc/shutdown` [ **-k** ] [ **-r** ] [ **-h** ] [ **-q** ] *time* [ *warning-message ...* ]

## DESCRIPTION

*shutdown* provides an automated shutdown procedure which a superuser can use to notify users nicely when the system is shutting down, saving them from system administrators, hackers, and gurus, who would otherwise not bother with niceties.

*time* is the time at which *shutdown* will bring the system down and may be the word **now** (indicating an immediate shutdown) or specify a future time in one of two formats: `+number` and `hour:min`. The first form brings the system down in *number* minutes and the second brings the system down at the time of day indicated (as a 24-hour clock).

At intervals which get closer together as apocalypse approaches, warning messages are displayed at the terminals of all users on the system. Five minutes before shutdown, or immediately if shutdown is in less than 5 minutes, logins are disabled by creating `/etc/nologin` and writing a message there. If this file exists when a user attempts to log in, *login*(1) prints its contents and exits. The file is removed just before *shutdown* exits.

At shutdown time a message is written in the file `/usr/adm/shutdownlog`, containing the time of shutdown, who ran shutdown and the reason. Then a terminate signal is sent at *init* to bring the system down to single-user state. Alternatively, if **-r**, **-h**, or **-k** was used, then *shutdown* will exec *reboot*(8), *halt*(8), or avoid shutting the system down (respectively). (If it isn't obvious, **-k** is to make people *think* the system is going down!)

The time of the shutdown and the warning message are placed in `/etc/nologin` and should be used to inform the users about when the system will be back up and why it is going down (or anything else).

If a remote host has mounted an NFS partition from this host, the shutdown message is also broadcast to that remote system via *rwall*(1). The **-q** option suppresses these messages from being sent to the remote system.

## FILES

`/etc/nologin` tells login not to let anyone except a superuser log in.  
`/etc/rmtab` list of remote hosts that have mounted this host.  
`/usr/adm/shutdownlog` log file for successful shutdowns.

## SEE ALSO

*login*(1), *rwall*(1), *wall*(1), *reboot*(8)

## BUGS

Only allows you to kill the system between now and 23:59 if you use the absolute time for shutdown.

## NOTES

*NFS* is an optional product; for more information, contact your CONVEX sales representative.

**NAME**

*slattach* - attach serial lines as network interfaces

**SYOPSIS**

```
/etc/slattach ttyname [ { + | - } { c | e | i } src-name dst-name [ baudrate ]
```

**DESCRIPTION**

*slattach* is used to assign a *tty* line to a network interface, and to define the network source and destination addresses. The *ttyname* parameter is a string of the form "ttyXX," or "/dev/ttyXX." *src-name* and *dst-name* are host names as found in */etc/hosts*. The optional *baudrate* parameter is used to set the speed of the connection. If not specified, the default of 9600 is used. Only the super-user may attach a network interface.

To detach the interface, kill the *slattach* process.

**FLAGS**

The following flags are valid (+ enables, - disables):

**e** - Enable decompression at input, even if **c** is not set.

**c** - Compress TCP headers on output, decompress headers on input.

**i** - Drop ICMP packets

**EXAMPLES**

```
/etc/slattach myhost remotehost tty08  
/etc/slattach myhost remotehost /dev/tty01 4800
```

**DIAGNOSTICS**

Messages indicate if the specified interface does not exist, the requested address is unknown, or the user is not privileged.

**SEE ALSO**

netstat(1), intro(4n), ifconfig(8), rc(8)

**NAME**

*spu* - transfer files between CONVEX UNIX and SPU UNIX file systems

**SYNOPSIS**

***spu*** [ **-r** ] [ **-w** ] *file*

**DESCRIPTION**

*spu* is used to transfer files between the SPU UNIX file system and the CONVEX UNIX file system. If the argument **-r** is supplied, the SPU UNIX file *file* will be written to the standard output of *spu*. If the argument **-w** is used, the standard input of *spu* will be written to the SPU UNIX file *file*. A full pathname should be used to specify the SPU UNIX file *file*. Only the superuser may use the **-w** argument with *spu*.

**SEE ALSO**

spucmd(8)

**EXAMPLES**

The following are common uses of *spu*.

<i>spu</i> -r /mnt/errlog	cat the error log from SPU
<i>spu</i> -r /mnt/errlog > foo	copy the error log from SPU to foo
<i>spu</i> -w /ioconfig < ./ioconfig	copy new ioconfig to SPU
<i>spu</i> -w /mnt/os/errlog.bu < /dev/null	truncate SPU file "errlog.bu"

## NAME

`spucmd` – execute single SPU UNIX command from CONVEX UNIX

## SYNOPSIS

`spucmd command_string`

## DESCRIPTION

`spucmd` is used to execute a single shell command under SPU UNIX from a CONVEX UNIX shell. The command string can be a single command such as "`ls`" or a single command with arguments such as "`ls -aC`" or "`ls -aC /mnt/os`". Multiple shell commands may be entered with one command if the commands are separated by semi-colons and the entire command string is enclosed in quotes. The shell filename generation operators `*`, `?`, and `./.` are not supported by `spucmd`.

A copy of the command being executed is output as each command is executed by the SPU shell. This is done by using the `-x` option to `sh(1)`. After the command is executed, any output from the shell will be output from `spucmd`. If the SPU shell command has a non-zero exit code, `spucmd` will terminate via an `exit(-1)`; otherwise it will terminate with an `exit(0)`.

Only the superuser may use this command. There is no way for `spucmd` to send data to the command being executed, but input and output can be redirected to local SPU files. `spucmd` will allow any command to be executed on SPU UNIX. Some command under SPU UNIX can cause CONVEX UNIX execution to be inhibited or stopped abruptly. Extreme caution is advised when executing commands on the SPU.

Used in conjunction with the `spu(8)` program, `spucmd` can accomplish many useful tasks for the system manager.

`spucmd` executes commands with a working directory of `/mnt/os`.

## SEE ALSO

`spu(8)`

## EXAMPLES

The following are common uses of `spucmd`.

<code>spucmd rm /mnt/errlog.bu</code>	remove the old <code>/mnt/errlog</code> overflow file
<code>spucmd df</code>	determine SPU disk space available
<code>spucmd mv /ioconfig /ioconfig.old</code>	move <code>/ioconfig</code> file to <code>/ioconfig.old</code>
<code>spucmd "cd /mnt/os ; ls -aC"</code>	list file in directory <code>/mnt/os</code>
<code>spucmd margin -v</code>	list voltage/clock reading of machine
<code>spucmd "date &gt; /date.out"</code>	save date in SPU file <code>/date.out</code>
<code>spucmd "date 8801072010"</code>	set SPU date to Jan 7, 1988 20:10

## RESTRICTIONS

`Spucmd` times out after a certain amount of time. So commands that wait for I/O, like `tail -f`, may not work.

**NAME**

statd, rpc.statd – network status monitor

**SYNOPSIS**

*/etc/rpc.statd*

**DESCRIPTION**

*statd* is an intermediate version of the status monitor. It interacts with *lockd(8c)* to provide the crash and recovery functions for the locking services on NFS.

**FILES**

*/etc/sm/*  
*/etc/sm.bak/*  
*/etc/state*

**SEE ALSO**

*lockd(8C)*, *statmon(5)*

**BUGS**

The crash of a site is only detected upon its recovery.

**NAME**

sticky - file mode sticky bit

**DESCRIPTION**

Setting the "sticky bit," mode 01000 (see *chmod(2)*), on a directory makes it impossible for a file in that directory to be unlinked by anyone other than its owner (or the superuser), even if that directory is publicly writable.

**NOTES**

Since the owner-unlink feature is enforced by the local filesystem, it may be unavailable on filesystems mounted via NFS from machines from other vendors or from Convex machines running older versions of ConvexOS.

Setting this bit on a Convex executable file has no impact on the execution of that program; the old use for this bit (saving text images on the swap volume) is no longer needed due to the Convex virtual memory system design.

## NAME

sumscripts – accounting summary awk scripts

## SYNOPSIS

```

/etc/connecttime | awk -f connecttime.awk
/etc/diskuse | awk -f diskbygrp.awk
/etc/diskuse | awk -f diskbyusr.awk
/etc/diskuse | cat - diskuse.out.old | awk -f diskmerge.awk
awk -f genbyact.awk general-format-logfile
awk -f genbygrp.awk general-format-logfile
awk -f genbygrpact.awk general-format-logfile
awk -f genbyusr.awk general-format-logfile
sa -g | awk -f sabyact.awk
sa -g | awk -f sabygrp.awk
awk -f tape.awk /usr/adm/tp-acct

```

## DESCRIPTION

The directory */usr/adm/sumscripts* contains the *awk* scripts listed above. The scripts summarize accounting data that is kept in various places throughout the system. Further summarization can be accomplished by piping the output of these scripts through *sort* and *idtoname*.

There are several summary possibilities that are not explicitly provided for. For example, there is no script to summarize data in general log file format by TTY or by user ID-activity ID. The large number of summary possibilities makes it impossible to provide every possible summary script. The scripts which are provided serve as a basis for customized scripts to be created by users as needed.

Many of these scripts either produce as output or accept as input a general log file format. This general log file format allows the same summary scripts to be used for different kinds of data.

**DEFINITION:** Files are in general log file format if they consist of newline-separated records with space-separated fields, and if each record contains the following fields in the order they are listed:

<i>time</i>	A time for the beginning of an event as returned by <i>time(3)</i> .
<i>amount</i> or <i>length</i>	A number whose meaning depends on the log file. For example, in <i>lpd-acct</i> , this field is the number of pages printed.
<i>uid</i>	User ID.
<i>gid</i>	Group ID.
<i>aid</i>	Activity ID.
<i>device</i>	A device, i.e. TTY, printer, or tape unit.
[ <i>other</i> ]	This optional field contains information which is specific to each log file. The scripts which operate on files that are in general log file format echo this information without doing any processing on it.

The printer log */usr/adm/lpd-acct* is in general log file format. The tape allocation log */usr/adm/tp-acct* may be converted to general log file format by using the *awk* script *tape.awk*. The log files */usr/adm/wtmp* and */usr/adm/bill-acct* may be combined into a single file in general log file format by running the *connecttime* utility and piping its output through the *connecttime.awk* script.

A description of each summary accounting *awk* script is given below.

<i>connecttime.awk</i>	Converts the output of <i>connecttime</i> to general log file format.
<i>diskbygrp.awk</i>	Summarizes the output of <i>diskuse</i> by group.
<i>diskbyusr.awk</i>	Summarizes the output of <i>diskuse</i> by user.
<i>diskmerge.awk</i>	Takes two versions of <i>diskuse</i> output and merges them together, creating totals for each user and group. The resulting output is in a format consistent with the output of <i>diskuse</i> , so that the same data can be run through <i>diskmerge.awk</i> repeatedly. The typical use of this script is to keep track of disk usage over time by taking old disk usage data and adding current data to it. The units produced are in kilobyte-time. For example, if this was done once a week, then the resulting data would be in units of kilobyte-weeks.
<i>genbyact.awk</i>	Summarizes data in general log file format by activity.
<i>genbygrp.awk</i>	Summarizes data in general log file format by group.
<i>genbygrpact.awk</i>	Summarizes data in general log file format by group-activity.
<i>genbyusr.awk</i>	Summarizes data in general log file format by user.
<i>sabyact.awk</i>	Summarizes the output of “ <i>sa -g</i> ” by activity.
<i>sabygrp.awk</i>	Summarizes the output of “ <i>sa -g</i> ” by group.
<i>tape.awk</i>	Converts from tape log format ( <i>/usr/adm/tp-acct</i> ) to general log file format.

#### EXAMPLE

```
awk -f /usr/adm/sumscripts/genbygrp.awk /usr/adm/lpd-acct | idtoname -g
```

The *awk* script */usr/adm/sumscripts/genbygrp.awk* will summarize the line printer usage data in */usr/adm/lpd-acct(5)* by group ID. This output is then piped through *idtoname(1)* to convert the group IDs in the first field to group names. Sample output is given below.

```
zero      60.000
hardware  29.000
test      352.000
os        2.000
```

#### SEE ALSO

*idtoname(1)*, *sort(1)*, *lpd-acct(5)*, *tp-acct(5)*, *connecttime(8)*, *diskuse(8)*, *pac(8)*, *sa(8)*, “Accounting” chapter in the *CONVEX System Manager’s Guide*.

**NAME**

swapon - specify additional device for paging and swapping

**SYNOPSIS**

```
/etc/swapon -a  
/etc/swapon name ...
```

**DESCRIPTION**

*Swapon* is used to specify additional devices on which paging and swapping are to take place. The system begins by swapping and paging on only a single device so that only one disk is required at bootstrap time. Calls to *swapon* normally occur in the system multi-user initialization file */etc/rc* making all swap devices available, so that the paging and swapping activity is interleaved across several devices.

Normally, the **-a** argument is given, causing all devices marked as "swap" swap devices in */etc/fstab* to be made available.

The second form gives individual block devices as given in the system swap configuration table. The call makes only this space available to the system for swap allocation.

**SEE ALSO**

swapon(2), init(8)

**FILES**

*/dev/d??b* normal paging devices

**BUGS**

There is no way to stop paging and swapping on a device. It is therefore not possible to make use of devices which may be dismounted during system operation.

**NAME**

`sync` – update the super block

**SYNOPSIS**

`/bin/sync [-v]`

**DESCRIPTION**

*Sync* executes the *sync* system primitive. *Sync* can be called to ensure all disk writes have been completed before the processor is halted, in a way not suitably done by *reboot(8)* or *halt(8)*.

See *sync(2)* for details on the system primitive.

The *-v* switch causes output to be produced to verify the execution of *sync*.

**SEE ALSO**

*sync(2)*, *fsync(2)*, *halt(8)*, *reboot(8)*, *update(8)*

**NAME**

**sysex** – system exerciser that simulates stressful load conditions

**SYNOPSIS**

**sysex**

or

**sysex functional-unit count ...**

**DESCRIPTION**

The system exerciser provides a convenient method of generating a system load that is at least equivalent to a full complement of users running a variety of tasks that are I/O and computation intensive.

**Sysex** can be used in either a *default* or a *specific* mode. The *default* mode can be specified by simply entering **sysex**, causing one iteration of a default exercise to be performed.

As an alternative, individual functional units can be exercised by explicitly including them in the command. Multiple iterations are specified by **sysex functional-unit count**, where *count* is the number of iterations desired. This is the *specific* mode. The functional units that are currently supported are:

**memory, tty, os, cpu, tape1, tape2, ..., tape16, disk, disk1, disk2, ..., disk16, hsp.**

Note that **tape[1-16]** and **disk[1-16]** refer to the different tape and disk drives attached to the computer, while **disk** refers to the disk subsystem independent of the number of drives attached. Specify one of the **disk[1-16]** functional units to ensure that a particular drive is exercised.

You must always indicate the number of iterations when using the *specific* mode. As an example,

**sysex memory 25 cpu 50 disk1 75 tape1 75**

will exercise the corresponding functional units for the specified number of iterations.

The *default* mode exercises a basic configuration consisting of one tape drive and one disk drive. This configuration is defined in a script called **default**, which can be modified to exercise a particular mix of peripherals. This script should be checked before using the *default* mode to ensure that the desired default action will occur.

**PREPARATION**

Be sure that

- 1) your current working directory is the exerciser directory **/sys/test/sysex**
- 2) you are logged in as **root**
- 3) the peripherals you want to exercise are present.

No other preparation is necessary unless you are going to exercise the **tape1, tape2, disk, or hsp** functional units. Each tape drive to be exercised should have a scratch tape (with a write-ring) mounted on it, and the drive should be on-line. The **/tmp** partition should have room for at least 7000K bytes before **disk** is used. For exercising the **hsp** functional unit, an FSE board set must be installed in the HIA. Refer to the CONVEX HIA USER'S GUIDE (081-000111-000). **Hsp** is an optional product; for more information, contact your CONVEX sales representative.

**OPERATIONS and RESULTS**

All execution is placed in background mode, except for the tty portion, which must be run in the foreground. There should be at least two terminals attached to the machine. The tty exercise should be performed on one terminal while another is used for the other exercises.

When an error is encountered, a description of the problem will be placed in a specified file, and the exercising of that functional unit is terminated. Otherwise, no output is generated except for a message after each iteration of a particular exercise.

**EXERCISER COMPONENTS**

The functionality of the CPU is exercised with a collection of computationally intensive routines performing mathematical calculations, using both scalar and vector techniques. Trigonometric and logarithmic functions are exercised over a wide range of values, and are calculated by both iteration and series expansion for comparison purposes.

Memory exercising is accomplished by a program that generates memory accesses across the entire logical address space. Blocks of memory throughout the address space are initialized to a particular value, and then accessed to make sure that the memory actually *remembered* the value.

Disk drives are exercised in two ways. One (**disk[1-16]**) involves reading random sectors from a drive, the other (**disk**) with the copying of a very large directory to another disk partition. Tape drives are exercised in a manner very similar to the disk drives. Large amounts of tape I/O will be generated, and the functions unique to tape drives will be exercised, including moving backwards and forwards between blocks, rewinding and moving to the end-of-file mark.

The UNIX operating system will be exercised by a set of programs that utilize system calls dealing with interprocess communication, I/O operations and file system manipulation. The tty interface will be exercised by moving large amounts of I/O through the tty subsystem.

**BUGS**

A copy of the system exerciser must be made for each person using it.

Only one **disk** exercise can be performed at a time, even when multiple copies of the system exerciser are created. The **disk[1-16]** exercises are not affected; that is,

```
sysex disk1 50 disk2 50
```

is correct usage.

Specifying the same functional unit more than once with the same copy of the exerciser is not handled correctly. For example,

```
sysex cpu 5 cpu 10
```

causes the two exercises to interfere with one another. This problem would still occur if they were separated into two commands.

**FILES**

```
/sys/test/sysex          the directory containing the exerciser.
```

**NAME**

`sysgen` - tool to aid in building customized operating system images

**SYNOPSIS**

`sysgen [-b] sysgen_file`

**DESCRIPTION**

`sysgen` uses the specified configuration file (*sysgen\_file*) and other input files to generate makefiles and include files which can be used to build tailored versions of the *vmunix*, *hsp*, *iop*, and *viop* operating system images for the specified configuration. It allows sites who have a ConvexOS source license to customize their systems to better meet the needs of their own site. Sites which only have a binary license can also use `sysgen` but have less freedom in the customization of their systems. `sysgen` can be used to add user written device drivers to either source or binary systems.

In addition to the configuration file, `sysgen` also uses a set of files to determine which source files are needed to generate a system. These input files can be augmented by a configuration specific set of files that give alternate files for a specific machine (see the FILES section below).

`sysgen` should be run from the `sysgen` subdirectory of the system source (usually `/sys/sysgen`). Typically, system managers use capitalized system names, like GOLLUM, as the names for *sysgen\_files*. `sysgen` will create the `../sysgen_file`, `../sysgen_file_hsp`, `../sysgen_file_iop`, and `../sysgen_file_viop` directories (if they don't already exist). These directories are used as build directories when running the *Makefiles* which `sysgen` creates.

The output of `sysgen` consists of a number of files which are placed in the directories mentioned above. These files are: *Makefiles*, files used by `make(1)` to build the system; a set of header files which contain the number of various devices that will be compiled into the system (*device\_name.h*); and a set of swap configuration files which contain definitions for the disk areas to be used for the root file system and for swapping.

After running `sysgen`, it is necessary to run "make depend" in each of the directories where the new makefiles were created. After running "make depend" you should run "make" in each directory in order to actually build the operating system images.

If you get any error messages from `sysgen`, you should fix the problems in your configuration file and try again. If you try to compile a system that had configuration errors, you will likely meet with failure.

As distributed with ConvexOS 8.1 and above, `sysgen` uses different compiler versions than previous versions of `sysgen`. To use the new release of `sysgen` under versions of ConvexOS prior to 8.1, the `-b` flag must be specified.

**FILES**

<code>/sys/sysgen/makefile.object</code>	generic makefile for <i>vmunix</i> image
<code>/sys/sysgen/makefile.ccu</code>	generic makefiles for CCU images
<code>/sys/sysgen/files.ccu</code>	list of CCU specific files
<code>/sys/sysgen/files.ccu.c?</code>	list of CPU specific CCU specific files
<code>/sys/sysgen/devices.c?</code>	name to major device mapping files
<code>/sys/sysgen/pseudo_devices</code>	list of valid pseudo-devices
<code>/sys/sysgen/controllers</code>	list of valid controller names and attributes
<code>/sys/sysgen/units</code>	list of valid unit names and attributes
<code>/sys/sysgen/GOLLUM</code>	system GOLLUM specification file
<code>/sys/GOLLUM</code>	CPU build directory
<code>/sys/GOLLUM_iop</code>	IOP build directory
<code>/sys/GOLLUM_hsp</code>	HSP build directory
<code>/sys/GOLLUM_viop</code>	VIOP build directory

**SEE ALSO**

See the *CONVEX System Generation Guide* for an example of a system build. See the *CONVEX Guide to User Written Device Drivers* for more information about adding device drivers to a system.

#### RESTRICTIONS

*Sysgen* causes the resulting version number of */vmunix* to be set to zero. The *vers(1)* command should be used to set a new version number.

#### NOTES

In ConvexOS 9.1 and above, "make depend" in the CPU build directory will issue several warnings about non-existent *\*.dep* files. These warnings can be ignored.

Also in ConvexOS 9.1 and above, the */sys/sysgen/files* file has been made obsolete. To add non-standard code to *vmunix* (such as user written device drivers), please refer to the *ConvexOS V9.1 Release Notice*.

#### BUGS

The line numbers reported in error messages are usually off by one.

## NAME

syslogd – log systems messages

## SYNOPSIS

```
/usr/etc/syslogd [ -fconfigfile ] [ -mmarkinterval ] [ -d ]
```

## DESCRIPTION

*syslogd* reads and logs messages into a set of files described by the configuration file */etc/syslog.conf*. Each message is one line. A message can contain a priority code, marked by a number in angle braces at the beginning of the line. Priorities are defined in *<sys/syslog.h>*. *syslogd* reads from the UNIX domain socket */dev/log* and from the Internet domain socket specified in */etc/services*.

*syslogd* configures when it starts up and whenever it receives a hangup signal. Lines in the configuration file have a *selector*, to determine the message priorities to which the line applies, and an *action*. The *action* field is separated from the selector field by one or more tabs or spaces.

Selectors are semicolon separated lists of priority specifiers. Each priority has a *facility* describing the part of the system that generated the message, a dot, and a *level* indicating the severity of the message. Symbolic names may be used. An asterisk selects all facilities. All messages of the specified level or higher (greater severity) are selected. More than one facility may be selected using commas to separate them. For example:

```
*.emerg;mail,daemon.crit
```

selects all facilities at the *emerg* level and the *mail* and *daemon* facilities at the *crit* level.

Known facilities and levels recognized by *syslogd* are those listed in *syslog(3)* without the leading "LOG\_". The additional facility "mark" has a message at priority LOG\_INFO sent to it every 20 minutes (this may be changed with the *-m* flag). The "mark" facility is not enabled by a facility field containing an asterisk. The level "none" may be used to disable a particular facility. For example,

```
*.debug;mail.none
```

sends all messages *except* mail messages to the selected file.

The second part of each line describes where the message is to be logged if this line is selected. There are four forms:

- A filename (beginning with a leading slash). The file will be opened in append mode.
- A hostname preceded by an at sign ("@"). Selected messages are forwarded to the *syslogd* on the named host.
- A comma separated list of users. Selected messages are written to those users if they are logged in.
- An asterisk. Selected messages are written to all logged-in users.

Blank lines and lines beginning with "#" are ignored.

For example, the configuration file:

```
kern,mark.debug      /dev/console
*.notice;mail.info   /usr/spool/adm/syslog
*.crit                /usr/adm/critical
kern.err              @ucbarpa
*.emerg               *
*.alert               eric,kridle
*.alert;auth.warning  ralph
```

logs all kernel messages and 20 minute marks onto the system console, all notice (or higher) level messages and all mail system messages except debug messages into the file */usr/spool/adm/syslog*, and all critical messages into */usr/adm/critical*; kernel messages of error severity or higher are forwarded to *ucbarpa*. All users are informed of any emergency messages; the users "eric" and "kridle" are informed of any alert messages; and the user "ralph" is informed of any alert message or any warning message (or higher) from the authorization system.

The flags are:

- f Specify an alternate configuration file.
- m Select the number of minutes between mark messages.
- d Turn on debugging.

*syslogd* creates the file */etc/syslog.pid*, if possible, containing a single line with its process id. This can be used to kill or reconfigure *syslogd*.

To bring *syslogd* down, it should be sent a terminate signal (e.g. kill `cat /etc/syslog.pid`).

#### FILES

<i>/etc/syslog.conf</i>	the configuration file
<i>/etc/syslog.pid</i>	the process id
<i>/dev/log</i>	name of the UNIX domain datagram log socket

#### SEE ALSO

logger(1), syslog(3)

## NAME

*syspic* - report system, disk, TTY, and network activity in windows

## SYNOPSIS

*syspic* [-p *picture*] [-f *picfile*] [-i *interval*] [-c *count*] [-a] [-u *unixbin*]

## DESCRIPTION

*syspic* is an interactive monitor that periodically reports statistics about system performance. Statistics are grouped into entities called "pictures" that consist of a few lines of information at the top of the screen and windows on the rest of the screen. Each window consists of a header and some statistics. *syspic* highlights numbers that are unusually large or small.

The command line switches, which may be given in any order, are:

- a This switch has meaning only when the picture chosen is *network*. Normally, daemons that listen for connections, such as the printer daemon, are not listed in the *network* picture. The -a switch causes these daemons to be listed.
- c *count* If *count* is positive, *syspic* automatically exits after *count* screen updates have been completed. If *count* is 0, *syspic* keeps updating until stopped. The default value for *count* is 0. Note that if *count* is 1, information that is computed as the change in some parameter since the previous update is not shown.
- f *picfile* Specifies a user-defined picture file. This is different from the -p switch in that the picture file need not reside in */usr/lib/syspic/\*.syspic*, so a user may have his or her own picture files that need not clutter up */usr/lib/syspic*.
- i *interval* Specifies how many seconds to wait between each screen update. Five seconds is a good choice, because some statistics within the system are gathered every five seconds. The default is five seconds.
- p *picture* Specifies which picture to display information about. Usual values for *picture* are *disk*, *disk1*, *disk2*, *kluster*, *network*, *proc*, *proc20*, *syscalls*, *system*, *tty*, and *vm*; the default picture is *system*. Any *.syspic* file that resides in directory */usr/lib/syspic* is treated as a picture and can be specified with the -p switch; this allows each site to have its own set of commonly-used pictures.
- u Specifies an alternate UNIX executable file. The default is */vmunix*.

The following keys are interpreted by *syspic*:

<SPACE>

<RETURN> Force *syspic* to update the screen and restart the timer.

<CTRL-L> Replot the screen.

- Freeze the screen. When the screen is frozen, no automatic updating will take place. *syspic* still interprets keys.

The *SIGINT* signal, usually generated by <CTRL-C> or <DEL>, is used to exit *syspic*. Additionally, *syspic* can be interrupted and later resumed with the *SIGTSTP* signal, which is typically generated by <CTRL-Z> from *cs*h.

The default set of pictures is:

*disk* The *disk* picture reports the activities of the first 32 currently-mounted disks (striped and non-striped), CCUs, CPU usage, and the size distribution of disk transfers. The disks are displayed in alphabetical order.

*disk1* This picture displays the same information as the *disk* picture, except instead of the size distribution of disk transfers, information on sixteen additional disks is

presented (for a total of 48 disks.)

- disk2* This is the same as the *disk1* picture, but displays information about disks 16 through 63. This is only useful for very large hardware configurations.
- kluster* This overview window reports statistics on various size distributions from disk transfers, pageins from a vnode, pageins from the swap device, and pageins from the zero fill device. Other data, such as memory usage, CPU usage, and buffered I/O are also displayed.
- network* When the *network* picture is first invoked, a list of connections to and from the host machine is displayed. New connections are added to the list and connections that have closed are deleted from it. If there is more than one page of connections, the number keys (0-9) may be used to select a page. The command-line switch **-a** may be used to specify that daemons listening for connections, such as the printer daemon, be included on the list of connections. A connection may be chosen for closer inspection by pressing the letter (a - o) that appears to the left of the connection's entry in the list of connections; **<ESCAPE>** returns back to the list of connections.
- proc* The top half of this picture contains information on processes, memory, and per-CPU usage. The *proc* picture is the only default picture that shows CPU usage broken down into a per-CPU basis; all other CPU usage windows are shown as calculated averages among all existing processors. The bottom half of this picture contains a display similar to *ps aux* (see the *ps(1)* command) for the top ten processes, sorted by percentage of CPU utilization.
- proc20* The *proc20* picture displays the top twenty processes, ranked by percentage of CPU utilization.
- syscalls* The *syscalls* picture shows execution counts for each of the 50 or so most frequently executed system calls and shows the total number of system calls per second. This picture is good for getting a breakdown of which system calls are contributing to an unusually high number of system calls and for looking at a single application's short-term or long-term system-call usage trend.
- system* The *system* picture is an overview of system activity which gives information about many phases of system performance including virtual memory, CPU, tape, CCU, disk, trap, and network I/O. A system manager can use this picture to get a feel for how the system is loaded and to watch for trouble spots.
- tty* The *tty* picture is intended to be used as a tool for evaluating the performance of a system's terminal subsystem. The system manager can use the information displayed by the *tty* picture to make intelligent decisions about which terminal controllers are the best candidates for high traffic and high baud rate terminal lines such as UUCP ports or laser printer ports. It can also be used to find "ringing" terminal ports (ports that are slowing the system down due to spurious characters generated by noise on a terminal line.) This picture initially displays information for the terminal lines belonging to controllers *ca0* through *ca3*. However, information for other controllers may be viewed by selecting another page of data. At the bottom of the screen, there is a message stating how to get to another page.
- vm* The *vm* picture gives an overview of various activities related to virtual memory, including faults, paging, buffered I/O, and a virtual memory summary window. With the introduction of multi-processor machines, the virtual memory system was largely redesigned, obscuring the fact as to whether physical I/O for virtual memory activities actually took place. The lower levels of the buffer cache now maintain the page cache, and the buffer cache has no idea whether an

I/O operation is being done for paging or for read/write system calls. The best general indicator of how the cache is working is the page hit rate, shown in both the *Buffered I/O* and the *VM Summary* windows.

In addition to the default set of pictures, users may define their own pictures. The best way to create a picture file is by mimicking one of the default picture files in */usr/lib/syspic/\*.syspic*. The following is a description of *.syspic* file format, **which is subject to change**:

The first line is optional; if present, it consists of the word "standard" which causes the picture to contain a standard set of information such as current time and load average (see below). This is followed by a line containing the picture name, followed by 0 or more lines consisting of a column number, row number, and window name, separated by a space. The upper left hand corner of the screen is designated as column 0, row 0; column and row are used to specify the upper left hand corner of a window. See the *RESTRICTIONS* section of this manual page for further information.

There are some entries in the default picture files with "Help" in their names. These entries are not really windows — they exist either for future use or to provide instructions about which keys to press.

Any window that is accessible only by selecting a network connection from the *Connections* window may not be used in a user-defined picture.

All default pictures display two standard lines at the top of the screen. These lines contain the following information:

<i>Current Time</i>	The last time a screen update occurred.
<i>Load Average</i>	Load averages averaged over the last 1 minute, 5 minutes, and 15 minutes. Load average is defined as the number of jobs in the run queue.
<i>Picture Title</i>	The title of the picture as it appears in the picture file. If there is sufficient space on the screen, the host name of the machine is prepended to the picture title.
<i>Reboot Time</i>	When the machine was last rebooted.
<i>Up</i>	Elapsed time since the last reboot.
<i>Users</i>	The number of users currently logged on.

In addition to the two lines at the top of the screen which are present in all default pictures, each of these pictures displays a set of windows in the lower area of the screen. The following table lists the windows that comprise each picture.

<i>disk</i>	<i>disk1</i>	<i>kluster</i>	<i>network</i>	<i>proc</i>
Buffered I/O	Buffered I/O	Buffered I/O	CCU Busy	Memory Mb.
CCU Busy	CCU Busy	CPU Usage	CPU Usage	Per-CPU Usage
CPU Usage	CPU Usage	DQ In	Connection	Processes
Calls	Calls	DQ Out	Connections	PS AU
Disk 0-15	Disk 0-15	Memory Mb.	Data Transfer	
DQ In	Disk 16-31	Paging	Mbuf Stats	
DQ Out	Processes	Processes	Network I/O	
Processes	Disk 32-47	Swap	Parameters	
Disk 16-31		Vnode	Recv. Sequence	
		Zfod	Retransmit	
			Send Sequence	
			TCP Stats	

## TCP Timers

<i>proc20</i>	<i>syscalls</i>	<i>system</i>	<i>tty</i>	<i>vm</i>
PS AU 20	System Calls	Buffered I/O	ca0	Buffered I/O
		CCU Busy	ca1	CPU Usage
		CPU Usage	ca2	Faults
		Faults	ca3	Memory Mb.
		Memory Mb.		Paging
		Network I/O		Processes
		Paging		Swap
		Path. Cache		VM Summary
		Processes		Vnode
		TTY Totals		Zfod
		Tape, Mb/s		

The *disk2* picture is identical to the *disk1* picture, but displays information about disks 16 through 63.

The windows are:

*Buffered I/O* This window shows parameters related to buffered user I/O and the buffer cache. **Mb/s** shows the average amount of buffered I/O in megabytes per second, measured by the kernel routine *rwip*. **Latency** is the average amount of time in milliseconds that a request to the buffered I/O system (e.g., *rwip*) took and is computed as time of completion minus time started. **Page Hits** is the hit ratio (from 0-100%) on the page cache. The page cache is a least-recently-used replaced cache containing both file system data and executable images. Each page is separately encached by file system and physical block address. The page cache effectively supercedes the buffer cache of previous releases. **Buffer Hits** is the hit ratio (from 0-100%) that a file system I/O request found the desired buffer pages already attached to a buffer header. Buffer headers are required by the device drivers for doing I/O. The desired pages may be found for an I/O request in the page cache (with a resulting page hit), even though they are not attached to a buffer header (resulting in a buffer miss). Buffer hits cannot occur without a corresponding page hit. **Bufcache** is the effective size of the buffer cache in megabytes. It is calculated by counting the number of pages attached to buffer headers. **Deleted Wr.** is the amount of file system data in megabytes that was thrown away because the file(s) that contained it were deleted before they were written to the disk. Temporary files are often not written to the disk at all. **Deferred Wr.** is the amount of file system data in megabytes that had been scheduled to be written to the disk but had the write deferred to some later time because another request for access to the data was made while the I/O request was on the disk queue.

*ca0, ca1, ca2, ..., caF*

Each "ca" window lists the teletypes connected to a separate ACM-001 or ACM-002 terminal controller. The number of characters input and output since the last screen update are displayed for each terminal line. The last line of each window, labeled "+", indicates the total number of characters input and output summed over all lines on the controller.

TTY lines that are not hooked up to the system have their data fields labeled with "-". These TTY lines do not have terminal controllers to hook up to and

thus cannot ever be active unless another terminal controller is added to the system's configuration.

Some TTY lines contain blank data fields. This indicates that this port has done no I/O since the system was last booted and differs from a value of 0, which means that the port has done I/O in the past but has done no I/O since the last screen update. A TTY line with a "-" in its data field has no terminal controller to interface to, whereas a TTY line with a blank data field has a terminal controller to interface to, but just happens not to have done any I/O since the system was last booted.

- Calls* Shows the number of system calls per second.
- CCU Busy* Shows what percent of the time each CCU (channel control unit) on the system is busy. Here, busy is determined by sampling at the CCU's clock interrupt frequency to see if any work is currently being done. For example, in an IOP this is determined by whether or not there is a task currently executing.
- Connection* Active only when a network connection has been chosen from the "Connections" window. This window is not like other windows because it does not have a label and a border; it consists solely of a selected connection's entry as it appears in the list of active connections. If a connection becomes closed, this line displays "CLOSED".
- Connections* List of active connections. Each connection's **local address**, **foreign address**, and **state** are listed. Also shown are number of bytes buffered in the receive and send queues for this connection, under the columns **recv Q** and **send Q**. When there is more than one page of connections, the number keys (0 - 9) may be used to move to another page. Pressing the letter that appears to the left of a connection yields another page of information which includes the connection-dependent windows **Data Transfer**, **TCP Timers**, **Send Sequence**, **Recv. Sequence**, and **Retransmit**. This page also contains other windows that are not connection-dependent.
- If a connection is closed, "CLOSED" appears in the **Connections** window, and no further updating is done of the connection-dependent windows. The other windows continue to update as usual because they do not rely on the connection's existence. Once a connection has been closed, it cannot be reopened.
- Restriction: a specific connection may not be selected from the *Connections* window in a user-defined picture.
- CPU Usage* A breakdown of percentage of usage of CPU time. **User** is the time spent running normal priority user processes. **User (n)** is the time spent running low priority ("niced") processes. **System** is the time spent running system code, and **Idle** is the amount of time that the CPU has nothing to do.
- Disk 0-15* Transfer rate and average latency of disk devices 0 through 15. The average latency, which has units of milliseconds, is computed as the time a request finished minus the time a request was originally queued. By definition, average latency includes the amount of time the request was in device queue. (However, since stripe devices are logical and not physical, there is no corresponding hardware device queue. Thus stripe devices don't have this additional time added in to their computation of latency.) The last line of the window, **tot**, shows the combined transfer rate of the disk devices shown in this window. Even though stripe devices have their transfer rates listed in the windows, they

are not included in each window's total. Since stripes are made up of physical drives, each byte would be counted twice if stripes were added to the total transfer rate: once for the physical disk and once for the stripe. High average latencies for all CCUs may mean that the system needs more CCUs. Uneven transfer rates or uneven average latencies may mean that there are enough CCUs, but work is being unevenly distributed between them.

- Disk 16-31* Exactly the same as the *Disk 0-15* window (see above), except this window shows information about disk devices 16 through 31.
- Disk 32-47* Exactly the same as the *Disk 0-15* window (see above), except this window shows information about disk devices 32 through 47.
- Disk 48-63* Exactly the same as the *Disk 0-15* window (see above), except this window shows information about disk devices 48 through 63.
- Data Transfer* Lists the number of bytes **Sent** and **Rcvd** (received) over the lifetime of the network connection.
- DQ In, DQ Out* The *DQ In* and *DQ Out* windows show the size distribution of disk transfers. The number of I/O requests (processed during the sample period) that are less than 4k bytes, as well as the number of requests for each size from 4k bytes through 64k bytes, are shown. The distribution shown in *DQ In* is the requests as they are sent to the disk driver. The disk driver coalesce adjacent requests into a single, larger request, which makes I/O more efficient. The resulting request-size distribution is shown in *DQ Out*.
- Faults* Trap/interrupt rate averages per second over the last 5 seconds. **Interrupts** is the number of interrupts per second not including the clock. **System Calls** is the number of system calls per second. **Context Sw** is the CPU context switch rate (switches/sec) for processes (**p**) and vectors (**v**).
- Mbuf Stats* Displays how many message buffers are in use in the system. **In use** shows the number of regular mbufs in use; **Free** is the number of regular mbufs on the free list. **Pages** represents the number of type 1 mbuf clusters in use, and **Free Pages** is the number of type 1 mbuf clusters on the free list. A regular mbuf is a 512-byte buffer. A type 1 cluster is an 8192-byte buffer. The sizes of regular and cluster mbufs are implementation-dependent and may change in future releases of CONVEX UNIX.
- Memory Mb.* Information about the usage of virtual and real memory. This information is displayed in megabytes. **Virt.** represents the total amount of virtual memory that might require paging to swap. **Real** memory is memory that is in use by processes and is actually in core (as opposed to memory that exists "virtually" on disk). **Free** memory is the amount of memory currently not in use.
- Network Buffers* Details how many message buffers are in use in the system. The number of buffers/pages in use and in existence is displayed. This window is no longer displayed on any of the default pictures, but similar information is available in the *Mbuf Stats* window.
- Network I/O* Shows activity taking place on local area networks attached to the system. Data is displayed for up to two network interfaces, two columns per interface. The numbers in the first column are accrued since the last screen update; the numbers in the second column are totals since the machine has been up. **In Packets** is the number of packets received over the network. **In Errors** is the number of receive errors. **Out Packets** is the number of packets transmitted over the network. **Out Errors** is the number of transmit errors. **Collisions** is the number of packet collisions that occur.

- Paging* Information about page faults, paging activity, and swaps. These are five-second averages and are displayed in units per second. **Reclaims** are pages that are freed but taken back by the process from which they were freed before being allocated anywhere else. Only pages paged to swap are included in this rate. Reclaims also include text pages that are given up by a process that exited but which are taken back by a new instance of the same program. **Page Ins** are faults that resulted in pages paged in, regardless of the number of pages or whether the pages were found in the cache or had to be read from the disk/network. **Page Outs** are the number of times the pager tried to page from a region, regardless of the number of pages paged out or whether the pages had to be written to the disk/network. **Pgin** is the rate of megabytes paged in per second as a result of "Page Ins" faults, regardless of whether the pages were found in the cache or had to be read in from the disk/network. **Pgout** is the rate of megabytes paged out per second as a result of "Page Outs" region scans, regardless of whether the pages had to be written to permanent storage. **Scan Rate** is the rate at which the pageout daemon is running through physical memory and freeing pages; also described as the number of megabytes of virtual memory scanned per second during "Page Outs" requests in order to produce "Pgout" megabytes of real memory. As the scan rate goes up, so does the number of pages freed (the number of reclaims will probably go up, too). Pages that are still free when the pageout daemon gets back to them are paged out. **Swap Ins** is the number of processes whose swap-priority-penalty has been removed. Processes are not actually swapped, but rather their priority is lowered for some time proportional to the amount of memory paged. **Swap Outs** is the number of processes partially swapped. These processes may still run, but their priority remains very low until their swap-priority-penalty is removed.
- Parameters* **Max seg size** is what this host believes is the maximum number of bytes a network packet can contain, not including the packet header. **Flags** contains other state information. If **Force** is set, the TCP software sends a byte even though it normally would not.
- Path. Cache* **Pathname Cache** shows the percentage of successful lookups ("hits") made to the pathname cache. **Calls/Hits** displays the number of calls versus hits to the pathname cache.
- Per-CPU Usage* Gives the same information as the *CPU Usage* window, except that here the information is shown with multiple columns, one for the total (or average) over all of the CPUs, followed by data on each individual processor in the system.
- Processes* **Runnable** processes are those trying to use the CPU. Processes in **Disk Wait** are awaiting the completion of disk (or other short-term I/O) operations, including paging and file requests. Processes in **Page Wait** are waiting for free memory pages, including memory for paging and file requests. **Sleeping** processes are those processes that are runnable or short sleepers (< 20 seconds). **Swapped** processes are those whose priority is lowered because they have been partially swapped.
- PS AU* The information in this window is very similar to output from the command *ps aux*, but it displays only the top 10 processes, ranked by "%CPU" (see *ps(1)*).
- PS AU 20* This is the same as *PS AU*, except it displays the top 20 processes.
- Recv. Sequence* **Wind** (window) is the number of bytes of buffering actually available on the receiving side of a network connection. **Next** is the expected sequence number of the next byte. **Urg** (urgent) is the expected sequence number of the next out-of-band byte. **Adv** (advertised window size) is sent from the receiver to the

- transmitter. It is the highest sequence number that the transmitter should send.
- Retransmit** **Rtt** (round-trip time) is the estimated round-trip time for the last packet sent on the network from this end of a connection. **Srtt** (smooth round-trip time) is computed as an average of the last few values of **Rtt** and is thus less susceptible to spikes. **Idle** is the elapsed time since the last packet was received. **Shift** controls how long to wait before retransmitting a packet. **Rtseq** is the sequence number for the packet currently being timed.
- Send Sequence** **Wind** (window) is what the transmitter of a network connection thinks the receiver's window size is. The sequence number of the next byte to transmit is given by **Next**. **Urg** (urgent) is the sequence number of the next out-of-band byte to transmit. **Max** is the highest sequence number the transmitter has ever sent. **Una** (unacknowledged) is the first sequence number the transmitter has sent that has not been acknowledged by the receiver. **Seq #** is the first sequence number of a segment. **Ack #** is the sequence number of the last packet which has been acknowledged.
- Swap** Shows the size distribution of pageins from the swap device. The sizes represented are less than 4k bytes, 4k, then increments of 4k up to 64k.
- System Calls** When the *syscalls* window is invoked, a sorted list is created that contains the 50 or so system calls that have been executed most often since the machine was last rebooted. Successive screen updates show two figures for each system call: how many times it has been executed since the last screen update, and how many times it has been executed since the machine was last rebooted. The totals shown are computed as the sum of the "normal" and "fast" system call totals. (Note: system call names are truncated to 10 characters.) Each screen update also shows the total number of system calls executed per second; this total takes into account all system calls, including those that do not have explicit listings in this window.
- Tape, Mb/s** The transfer rates of tape drives *ta0*, *ta1*, ... *ta5* are expressed in megabytes per second, along with the sum (**tot**) of these transfer rates. Fields with "-" indicate that, under the current system configuration, there is no tape drive hooked up in that slot.
- TCP Stats** **Bad Sum** is incremented when a received packet fails the internet checksum algorithm. **Hdr Drop** (header drop) is the number of times a packet was dropped because its header was too small. **Bad Off** is incremented when a packet is smaller than the size of a header or larger than the maximum packet size. **Duplicat** is the number of duplicate data packets received. **Retrans** is the number of data packets retransmitted.
- TCP Timers** When non-zero, these network connection timers are continuously decreasing. **REXMT** is the re-transmit timer. When the transmitter thinks the receiver's window size is zero, the transmitter periodically probes the receiver to make sure the window size is really zero. **PERSIST** is the amount of time remaining until the next probe. **KEEP** controls when this side of the connection sends data to the other side. This is used to determine whether or not the other machine is up. **2MSL** stands for "2 times max segment lifetime" and is used to delay final shutdown of a connection.
- TTY Totals** The number of characters input and output are displayed for each ACM-001 and ACM-002 terminal controller. These numbers represent the number of characters input and output since the last screen update, summed over all TTY lines hooked up to each controller. Fields with "-" indicate that, under the current system configuration, there is no TTY controller hooked up in that slot. The

final row, labeled "+", represents the sum of all ACM-001 and ACM-002 terminal controllers.

- VM Summary* The *VM Summary* window displays the following information per *syspic* sample period: number of page faults, number of write faults, number of PTEs copied as a result of a fork, number of pages copied as a result of a write fault, number of pages filled from a vnode, number of pages zero filled, number of pages filled from the swap device, number of pages written to a vnode, number of pages written to the swap device. The VM summary also displays the hit rate on the page cache; this rate includes both file and paging I/O.
- Vnode* Shows the size distribution of pageins from a vnode (i.e. regular file). Size notation is the same as for *DQ In* and *DQ Out*. The most common occurrences of these pageins are program text and data pages. The requests may have been satisfied by encached pages in which case no I/O was done, or it may have been read from the file.
- Zfod* Shows the size distribution of pageins from the zero-fill-on-demand device. The most common occurrence of a zero fill page fault is from the bss segment. A single fault often returns a cluster of pages, hopefully saving the time of additional faults.

## FILES

<i>/vmunix</i>	kernel name list
<i>/dev/mem</i>	kernel data values
<i>/lib/kernsyms/symdata_*</i>	kernel symbol addresses
<i>/usr/lib/syspic/*.syspic</i>	standard <i>syspic</i> display specification files
<i>/dev/r{d[adu],st}*ctl</i>	disk control devices used to obtain disk data

## NOTE

The execution of *syspic* itself can affect the statistics that *syspic* measures.

## RESTRICTIONS

Some information is computed as the change in some parameter since the screen was last updated. Information of this type is displayed beginning on the second screen update. As a result, if a *count* of 1 is specified using the *-c* switch, such information is blank upon exit of *syspic*.

*syspic* won't run on screens smaller than 24 rows by 80 columns.

Because of a restriction in the Maryland Window Library, *syspic* cannot use the 80th column on some terminals; anyone designing their own picture should make sure that their picture does not use the 80th column.

*syspic* uses entries in */etc/mstab* to locate mounted drives. These entries are used to build the name of the drives' control devices. Thus, the block-special devices are assumed to be in the */dev* directory and to have CONVEX-supported names.

## BUGS

The way CPU time is computed causes the number of interrupts reported in the *Faults* window to be reported as "-1" occasionally.

In the *network* picture, there is no way to see the *TCP Stats* window except by looking at a specific connection, even though the *TCP Stats* window shows information that is connection-independent.

*syspic* makes assumptions about device names that are not always valid.

Sometimes the output gets corrupted, particularly if keys are pressed while output is being displayed on the terminal. To clean up a terminal that is in a bad state, try replotting the screen or re-entering *syspic*.

**NAME**

talkd, in.talkd - talk daemon

**SYNOPSIS**

`/usr/etc/in.talkd [ -timeout ]`

**DESCRIPTION**

*talkd* is the server used by the *talk(1)* program. It is used to establish the connection between users and machines, perform the *talk* requests, and handle communications termination and errors.

*talkd* is invoked as needed by *inetd(8C)*, and times out if inactive for two minutes. The `-t` option can be specified to use a different timeout length than the default. The timeout length is specified in minutes.

**SEE ALSO**

*talk(1)*, *services(5)*, *inetd(8C)*

**BUGS**

The *talk* protocol is architecture dependent, and cannot be relied upon to work between CONVEX machines and machines of other makers.

## NAME

telnetd - DARPA TELNET protocol server

## SYNOPSIS

/etc/telnetd

## DESCRIPTION

*telnetd* is a server that supports the DARPA standard TELNET virtual terminal protocol. *telnetd* is invoked by the internet server (see *inetd(8)*), normally for requests to connect to the TELNET port as indicated by the */etc/services* file (see *services(5)*).

*telnetd* operates by allocating a pseudo-terminal device (see *pty(4)*) for a client, then creating a login process which has the slave side of the pseudo-terminal as *stdin*, *stdout*, and *stderr*. *telnetd* manipulates the master side of the pseudo-terminal, implementing the TELNET protocol and passing characters between the remote client and the login process.

When a TELNET session is started up, *telnetd* sends TELNET options to the client side indicating a willingness to do *remote echo* of characters, *suppress go ahead*, *remote flow control*, and to receive *terminal type*, *terminal speed*, and *window size* information from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The pseudo-terminal allocated to the client is configured to operate in "cooked" mode, and with XTABS and CRMOD enabled (see *tty(4)*).

*telnetd* is willing to do: *echo*, *binary*, *suppress go ahead*, and *timing mark*. *telnetd* is willing to have the remote client do: *binary*, *terminal type*, *terminal speed*, *window size*, *toggle flow control*, and *suppress go ahead*.

## SEE ALSO

telnet(1)

## BUGS

Some TELNET commands are only partially implemented.

Because of bugs in the original 4.2 BSD *telnet(1)*, *telnetd* performs some dubious protocol exchanges to try to discover if the remote client is, in fact, a 4.2 BSD *telnet(1)*.

*Binary mod:* has no common interpretation except between similar operating systems (UNIX, in this case).

The terminal type name received from the remote client is converted to lower case.

*telnetd* never sends TELNET *go ahead* commands.

## NAME

timed - time server daemon

## SYNOPSIS

```
/usr/etc/timed [-t] [-M] [-n network] [-i network]
```

## DESCRIPTION

*timed* is the time server daemon and is normally invoked at boot time from the *rc(8)* file. It synchronizes the host's time with the time of other machines in a local area network running *timed(8)*. These time servers will slow down the clocks of some machines and speed up the clocks of others to bring them to the average network time. The average network time is computed from measurements of clock differences using the ICMP timestamp request message.

The service provided by *timed* is based on a master-slave scheme. When *timed(8)* is started on a machine, it asks the master for the network time and sets the host's clock to that time. After that, it accepts synchronization messages periodically sent by the master and calls *adjtime(2)* to perform the needed corrections on the host's clock.

It also communicates with *date(1)* in order to set the date globally, and with *timedc(8)*, a timed control program. If the machine running the master crashes, then the slaves will elect a new master from among slaves running with the *-M* flag. A *timed* running without the *-M* flag will remain a slave. The *-t* flag enables *timed* to trace the messages it receives in the file */usr/adm/timed.log*. Tracing can be turned on or off by the program *timedc(8)*. *timed* normally checks for a master time server on each network to which it is connected, except as modified by the options described below. It will request synchronization service from the first master server located. If permitted by the *-M* flag, it will provide synchronization service on any attached networks on which no current master server was detected. Such a server propagates the time computed by the top-level master. The *-n* flag, followed by the name of a network which the host is connected to (see *networks(5)*), overrides the default choice of the network addresses made by the program. Each time the *-n* flag appears, that network name is added to a list of valid networks. All other networks are ignored. The *-i* flag, followed by the name of a network to which the host is connected (see *networks(5)*), overrides the default choice of the network addresses made by the program. Each time the *-i* flag appears, that network name is added to a list of networks to ignore. All other networks are used by the time daemon. The *-n* and *-i* flags are meaningless if used together.

## FILES

```
/usr/adm/timed.log      tracing file for timed
/usr/adm/timed.masterlog  log file for master timed
```

## SEE ALSO

*date(1)*, *adjtime(2)*, *gettimeofday(2)*, *icmp(4P)*, *timedc(8)*,  
*TSP: The Time Synchronization Protocol for UNIX 4.3BSD*, R. Gusella and S. Zatti

## NAME

timedc – timed control program

## SYNOPSIS

`/usr/etc/timedc [ command [ argument ... ] ]`

## DESCRIPTION

*timedc* is used to control the operation of the *timed* program. It may be used to:

- measure the differences between machines' clocks,
- find the location where the master time server is running,
- enable or disable tracing of messages received by *timed*, and
- perform various debugging actions.

Without any arguments, *timedc* will prompt for commands from the standard input. If arguments are supplied, *timedc* interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected causing *timedc* to read commands from a file. Commands may be abbreviated; recognized commands are:

? [ command ... ]

help [ command ... ]

Print a short description of each command specified in the argument list, or, if no arguments are given, a list of the recognized commands.

clockdiff host ...

Compute the differences between the clock of the host machine and the clocks of the machines given as arguments.

trace { on | off }

Enable or disable the tracing of incoming messages to *timed* in the file `/usr/adm/timed.log`.

quit

Exit from *timedc*.

Other commands are included for use in testing and debugging *timed*; the help command and the program source may be consulted for details. If the source is not available then these commands are of limited utility.

## FILES

`/usr/adm/timed.log` tracing file for *timed*

`/usr/adm/timed.masterloglog` file for master *timed*

## SEE ALSO

`date(1)`, `adjtime(2)`, `icmp(4P)`, `timed(8)`,

*TSP: The Time Synchronization Protocol for UNIX 4.3BSD*, R. Gusella and S. Zatti

## DIAGNOSTICS

?Ambiguous command abbreviation matches more than one command

?Invalid command no match found

?Privileged command command can be executed by root only

**NAME**

`tpconfig` – configure the tape system

**SYNOPSIS**

`tpconfig` [*command*]

**DESCRIPTION**

The system administrator runs this program to set up a database which keeps track of the available tape drives and their attributes. (The file `/etc/tapecap` is no longer used, and the tape attributes it controlled are now controlled by `tpconfig`.)

If no arguments are given, `tpconfig` acts like a shell and prompts for commands until exited. If arguments are given, `tpconfig` interprets the arguments as a single command and exits.

The following commands are recognized, and can be abbreviated to their minimum unambiguous length:

**Add ACcess\_drive** *Group\_set group\_list type:unit*

**Add ACcess\_drive** *User\_set user\_list type:unit*

These commands add users or groups to a set that determines who is allowed to access a particular tape drive.

The magic character “\*” represents all users or all groups. Users and groups can be specified by either their name in `/etc/passwd` or `/etc/group`, or by the numeric id surrounded by brackets (e.g., `[0]` is the same as `root`).

**Add ALlocate\_drive** *Group\_set group\_list type:unit*

**Add ALlocate\_drive** *User\_set user\_list type:unit*

These commands add users or groups to a set that determines who is allowed to allocate a particular tape drive without mounting a tape.

The magic character “\*” represents all users or all groups. Users and groups can be specified by either their name in `/etc/passwd` or `/etc/group`, or by the numeric id surrounded by brackets (e.g., `[0]` is the same as `root`).

**Add Bypass\_labels** *Group\_set group\_list type:unit*

**Add Bypass\_labels** *User\_set user\_list type:unit*

These commands add users or groups to a set that determines who is allowed to bypass label processing for a particular drive.

The magic character “\*” represents all users or all groups. Users and groups can be specified by either their name in `/etc/passwd` or `/etc/group`, or by the numeric id surrounded by brackets (e.g., `[0]` is the same as `root`).

**Add DRive** *type:unit* [*Nocontrol*] [*Timeout=N*]

Adds a new tape drive to the database. The drive is identified by the *type*, which can be any string that doesn't contain white space or a colon, and by the *unit* which is an integer unit number. Examples: `mt:0`, `exabyte:0`.

If *Nocontrol* is given, then this drive is not controlled by the tape system. By default, the drive is controlled by the tape system.

The *timeout* specifies how long, in minutes, a drive can be idle before it is taken from the user. The default timeout is 60 minutes.

**Add Label** *label\_type daemon*

Adds a new label type to the database. The label type is identified by the string *label\_type*. *daemon* is a full path to the daemon that handles this label type. The only label type currently supported is *ansi*, and the daemon is */usr/lib/tape/ansidaemon*.

**Add Node** *path* [*Speed=string*] [*Density=string*] [*Rewind*] [*Buffered*] *type:unit*

Adds a new node to a drive in the database. The node is identified by the complete path to the special file (e.g., */dev/rmt8*), and the drive is identified by the *type* and *unit*.

The speed is any string that describes the speed of this node (e.g., 200ips).

The density is any string that describes the density of this node (e.g., 6250).

If *Rewind* is specified, then this node is a rewind device. The default is no-rewind.

If *Buffered* is specified, then this node will use internal buffering. The default is unbuffered.

**Delete A***C*cess\_drive *Group\_set group\_list type:unit*

**Delete A***C*cess\_drive *User\_set user\_list type:unit*

These commands delete users or groups from a set that determines who is allowed to access a particular tape drive.

The magic character "\*" represents all users or all groups. Users and groups can be specified by either their name in */etc/passwd* or */etc/group*, or by the numeric id surrounded by brackets (e.g., *[0]* is the same as *root*).

**Delete A***L*locate\_drive *Group\_set group\_list type:unit*

**Delete A***L*locate\_drive *User\_set user\_list type:unit*

These commands delete users or groups from a set that determines who is allowed to allocate a particular tape drive without mounting a tape.

The magic character "\*" represents all users or all groups. Users and groups can be specified by either their name in */etc/passwd* or */etc/group*, or by the numeric id surrounded by brackets (e.g., *[0]* is the same as *root*).

**Delete B**ypass\_labels *Group\_set group\_list type:unit*

**Delete B**ypass\_labels *User\_set user\_list type:unit*

These commands delete users or groups from a set that determines who is allowed to bypass label processing for a particular drive.

The magic character "\*" represents all users or all groups. Users and groups can be specified by either their name in */etc/passwd* or */etc/group*, or by the numeric id surrounded by brackets (e.g., *[0]* is the same as *root*).

**Delete Drive** *type:unit*

Deletes a drive from the database. All nodes associated with this drive are also deleted.

**Delete Label** *label\_type*

Deletes a label type from the database.

**Delete Silohost** *<type:unit>*

Deletes the association of a drive to the server host of the StorageTek Automated Cartridge System, commonly referred to as a *tape silo*.

**Delete Node** *path*

Deletes a node from the database.

**SEt ACcess\_drive Group\_set** *group\_list type:unit*

**SEt ACcess\_drive User\_set** *user\_list type:unit*

These commands define the set of users or groups that determines who is allowed to access a particular tape drive. Users and groups can be specified by either their name in */etc/passwd* or */etc/group*, or by the numeric id surrounded by brackets (e.g., *[0]* is the same as *root*).

The magic character "\*" represents all users or all groups.

**SEt ALlocate\_drive Group\_set** *group\_list type:unit*

**SEt ALlocate\_drive User\_set** *user\_list type:unit*

These commands define the set of users or groups that determines who is allowed to allocate a particular tape drive without mounting a tape. Users and groups can be specified by either their name in */etc/passwd* or */etc/group*, or by the numeric id surrounded by brackets (e.g., *[0]* is the same as *root*).

The magic character "\*" represents all users or all groups. Users and groups can be specified by either their name in */etc/passwd* or */etc/group*, or by the numeric id surrounded by brackets (e.g., *[0]* is the same as *root*).

**SEt Bypass\_labels Group\_set** *group\_list type:unit*

**SEt Bypass\_labels User\_set** *user\_list type:unit*

These commands define the set of users or groups that determines who is allowed to bypass label processing for a particular drive.

The magic character "\*" represents all users or all groups. Users and groups can be specified by either their name in */etc/passwd* or */etc/group*, or by the numeric id surrounded by brackets (e.g., *[0]* is the same as *root*).

**SEt Control ON|OFF** *type:unit*

This command puts a drive under the control of the tape system.

**SEt Default Density** *density*

This command defines the default tape density. Unless the user requests a different tape density with the *tpmount* command, this tape density is used.

**SEt Default Drive** *type*

This command defines the default drive type. Unless the user requests a different type with the *tpmount* command, this type is used.

**SEt Default Flags** [*Rewind|Norewind*] [*Character|Block|Labeled*]

This command sets the default flags for *tpmount* requests. **Rewind** and **Norewind** select whether the device will or will not rewind on close. **Character**, **Block**, and **Labeled** select the default device mode.

**SEt Default Speed** *speed*

This command defines the default drive speed. Unless the user requests a different speed with the *tpmount* command, this speed is used.

**SEt No\_default Density**

This command undefines the default drive density.

#### **SEt No\_default Speed**

This command undefines the default drive speed.

#### **SEt Queueing Enabled|Disabled**

When queueing is enabled, all tape mount and drive allocation requests are passed to *opreq(1)* for an operator to handle or fulfill. Otherwise, drive allocations are handled by *tpdaemon* and tape mounts are not allowed (ie, the users must mount their own tapes). Warning: when this option is changed, the tape system must be idle before the change takes effect.

#### **SEt Silohost *hostname drive\_string type:unit***

This command associates this drive with a tape silo server host. *hostname* is the network name of the workstation serving the tape silo. *drive\_string* is the string that defines the location of the drive within the tape silo.

#### **SEt Timeout *N type:unit***

This command changes the timeout value for the specified drive.

#### **SHow All**

Shows everything.

#### **SHow DEfaults**

Shows all of the default values.

#### **SHow DRive *type:unit***

Shows all information associated with the indicated drive, including the nodes.

#### **SHow Labels**

Shows all of the available label types.

#### **SHow Node *path***

Shows all information associated with the indicated node.

#### **SNAPSHOT [*file*]**

The command creates an ASCII snapshot of the current configuration database. If *file* is not given, the snapshot is written to standard output. The snapshot consists of a set of *tpconfig* commands that will create a database equivalent to the current database.

#### **FILES**

*/usr/lib/tape/config.db*            tape configuration database

#### **SEE ALSO**

*silomount(1)*, *silodismount(1)*, *siloquery(1)*, *siloenter(1)*, *siloject(1)*, *tpmount(1)*, *tpunmount(1)*, *tpattr(1)*, *tplabel(1)*, *tpunlabel(1)*, *tpqueue(1)*, *tpwait(1)*, *tape(3)*, *mt(1)*, *ansidaemon(8)*, *tpdaemon(8)*

**NAME**

tpdaemon - tape daemon

**SYNOPSIS**

**/usr/lib/tape/tpdaemon** [-debug] [-syslog] [-off]

**DESCRIPTION**

*tpdaemon* is the main daemon for the tape subsystem. It is normally started from */etc/rc.local* without any of the options selected.

Options:

**-debug**

Enables logging of debug messages.

**-syslog**

Causes log messages to go to standard output instead of to syslog. This is only used for debugging purposes.

**-off**

Causes *tpdaemon* to merely rewind a tape when it would normally take the tape offline. This is only used for debugging purposes.

**FILES**

<b>/usr/lib/tape/config.db</b>	tape configuration database
<b>/usr/lib/tape/tpdaemon.lock</b>	tape daemon lock file

**SEE ALSO**

tpmount(1), tpunmount(1), tpattr(1), tplabel(1), tpunlabel(1), tpqueue(1), tpwait(1), tape(3), ansi-daemon(8), tpconfig(8), mt(1)

**NAME**

*trpt* - transliterate protocol trace

**SYNOPSIS**

*trpt* [-a] [-s] [-t] [-f] [-j] [-p hex-address] [system [core]]

**DESCRIPTION**

*trpt* interrogates the buffer of TCP trace records created when a socket is marked for "debugging" (see *setsockopt(2)*), 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.

- a in addition to the normal output, print the values of the source and destination addresses for each packet recorded.
- 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.
- f follow the trace as it occurs, waiting a short time for additional records each time the end of the log is reached. If *trpt* is killed with a *SIGTERM*, it will flush its output buffers before exiting.
- 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, the address of which follows.

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(1)*. Then run *trpt* with the *-p* option, supplying the associated protocol control block addresses. The *-f* option can be used to follow the trace log once the trace is located. 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. The

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**

/vmunix  
/dev/kmem

**SEE ALSO**

*setsockopt(2)*, *netstat(1c)*

**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 race record.

The output format is inscrutable and should be described here.

**NOTES**

*trpt* is an optional product; for more information, contact your CONVEX sales representative.

**NAME**

update - periodically update the super block

**SYNOPSIS**

**/etc/update**

**DESCRIPTION**

*Update* is a program that executes the *sync(2)* primitive every 30 seconds. This insures that the file system is fairly up to date in case of a crash. This command should not be executed directly, but should be executed out of the initialization shell command file.

**SEE ALSO**

*sync(2)*, *sync(8)*, *init(8)*, *rc(8)*

**BUGS**

With *update* running, if the CPU is halted just as the *sync* is executed, a file system can be damaged. A fix would be to have *sync(8)* temporarily increment the system time by at least 30 seconds to trigger the execution of *update*. This would give 30 seconds grace to halt the CPU.

## NAME

*uucico*, *uucpd* – transfer files queued by *uucp* or *uux*

## SYNOPSIS

```
/usr/lib/uucp/uucico [ -dspooldir ] [ -ggrade ] [ -rrole ] [ -R ] [ -ssystem ] [ -xdebug ] [ -L ]
[ -tturnaround ]
/etc/uucpd
```

## DESCRIPTION

*uucico* performs the actual work involved in transferring files between systems. *uucp*(1C) and *uux*(1C) merely queue requests for data transfer that *uucico* processes.

The following options are available.

**-dspooldir**

Use *spooldir* as the spool directory. The default is */usr/spool/uucp*.

**-ggrade** Only send jobs of grade *grade* or higher on this transfer. The grade of a job is specified when the job is queued by *uucp* or *uux*.

**-rrole** *role* is either 1 or 0; it indicates whether *uucico* is to start up in master or slave role, respectively. 1 is used when running *uucico* by hand or from *cron*(8). 0 is used when another system calls the local system. Slave role is the default.

**-R** Reverse roles. When used with the **-r1** option, this tells the remote system to begin sending its jobs first, instead of waiting for the local machine to finish.

**-ssystem**

Call only system *system*. If **-s** is not specified, and **-r1** is specified, *uucico* attempts to call all systems for which there is work. If **-s** is specified, a call is made even if there is no work for that system. This is useful for polling.

**-xdebug** Turn on debugging at level *debug*. Level 5 is a good start when trying to find out why a call failed. Level 9 is very detailed. Level 99 is absurdly verbose. If *role* is 1 (master), output is normally written to the standard message output *stderr*. If *stderr* is unavailable, output is written to */usr/spool/uucp/AUDIT/system*. When *role* is 0 (slave), debugging output is always written to the AUDIT file.

**-L** Only call "local" sites. A site is considered local if the device-type field in *L.sys* is one of LOCAL, DIR or TCP.

**-tturnaround**

Use *turnaround* as the line turnaround time (in minutes) instead of the default 30. If *turnaround* is missing or 0, line turnaround is disabled. After *uucico* has been running in slave role for *turnaround* minutes, it attempts to run in master role by negotiating with the remote machine. In earlier versions of *uucico*, a transfer of many large files in one direction would hold up mail going in the other direction. With the turnaround code working, the message flow is more bidirectional in the short term. This option only works with newer *uucico*'s and is ignored by older ones.

If *uucico* receives a SIGFPE (see *kill*(1)), it toggles the debugging on or off.

*uucpd* is the server for supporting UUCP connections over networks. *uucpd* listens for service requests at the port indicated in the "uucp" service specification; see *services*(5). The server provides login name and password authentication before starting up *uucico* for the rest of the transaction.

*uucico* is commonly used either of two ways: as a daemon run periodically by *cron*(8) to call out to remote systems, and as a "shell" for remote systems who call in. For calling out periodically, a typical line in *crontab* would be:

```
0 * * * * /usr/lib/uucp/uucico -r1
```

This runs *uucico* every hour in master role. For each system that has transfer requests queued, *uucico* calls the system, logs in, and executes the transfers. The file *L.sys*(5) is consulted for information about how to log in, while *L-devices*(5) specifies available lines and modems for calling.

For remote systems to dial in, an entry in the *passwd*(5) file must be created, with a login "shell" of *uucico*. For example:

```
nuucp:Password:6:1::/usr/spool/uucppublic:/usr/lib/uucp/uucico
```

The UID for UUCP remote logins is not critical, as long as it differs from the UUCP Administrative login. The latter owns the UUCP files, and assigning this UID to a remote login would be an extreme security hazard.

## FILES

/usr/lib/uucp/	UUCP internal files/utilities
/usr/lib/uucp/L-devices	Local device descriptions
/usr/lib/uucp/L-dialcodes	Phone numbers and prefixes
/usr/lib/uucp/L.aliases	Hostname aliases
/usr/lib/uucp/L.cmds	Remote command permissions list
/usr/lib/uucp/L.sys	Host connection specifications
/usr/lib/uucp/USERFILE	Remote directory tree permissions list
/usr/spool/uucp/	Spool directory
/usr/spool/uucp/AUDIT/*	Debugging audit trails
/usr/spool/uucp/C./	Control files directory
/usr/spool/uucp/D./	Incoming data file directory
/usr/spool/uucp/D.hostname/	Outgoing data file directory
/usr/spool/uucp/D.hostnameX/	Outgoing execution file directory
/usr/spool/uucp/CORRUPT/	Place for corrupted C. and D. files
/usr/spool/uucp/ERRLOG	UUCP internal error log
/usr/spool/uucp/LOGFILE	UUCP system activity log
/usr/spool/uucp/LCK/LCK..*	Device lock files
/usr/spool/uucp/SYSLOG	File transfer statistics log
/usr/spool/uucp/STST/*	System status files
/usr/spool/uucp/TM./	File transfer temp directory
/usr/spool/uucp/X./	Incoming execution file directory
/usr/spool/uucppublic	Public access directory

## SEE ALSO

uucp(1C), uuq(1C), uux(1C), L-devices(5), L-dialcodes(5), L.aliases(5), L.cmds(5), L.sys(5), uuclean(8C), uupoll(8C), uusnap(8C), uuxqt(8C)

D. A. Nowitz and M. E. Lesk, *A Dial-Up Network of UNIX Systems*.

D. A. Nowitz, *Uucp Implementation Description*.

**NAME**

uuclean - uucp spool directory clean-up

**SYNOPSIS**

`/usr/lib/uucp/uuclean [-m] [-ntime] [-ppre] [-dsubdirectory]`

**DESCRIPTION**

*uuclean* scans the spool directory for files with the specified prefix and deletes all those that are older than the specified number of hours.

The following options are available:

**-ppre** Scan for files with *pre* as the file prefix. Up to 10 **-p** arguments may be specified.

**-ntime** Files whose age is more than *time* hours are deleted if the prefix test is satisfied. (Default *time* is 72 hours.)

**-m** Send mail to the owner of the file when it is deleted.

**-dsubdirectory**

Only the specified subdirectory is cleaned.

This program will typically be run daily by *cron*(8).

**FILES**

`/usr/spool/uucp` Spool directory

**SEE ALSO**

`uucp(1C)`, `uux(1C)`, `uucico(8C)`

**NAME**

uulook - monitor inter-system communications

**SYNOPSIS**

**uulook**

**DESCRIPTION**

*Uulook* displays the end of the *uucp* log. It then monitors it, printing any new entries.

*Uulook* is a one-line shell script:

```
sniff -600 /usr/spool/uucp/LOGFILE
```

**FILES**

/usr/spool/uucp/LOGFILE

**SEE ALSO**

sniff(1), tail(1), uucp(1c)

## NAME

uupoll - poll a remote UUCP site

## SYNOPSIS

uupoll [ *-ggrade* ] [ *-n* ] *system*

## DESCRIPTION

*uupoll* is used to force a poll of a remote system. It queues a null job for the remote system and then invokes *uucico*(8C).

The following options are available:

*-ggrade* Only send jobs of grade *grade* or higher on this call.

*-n* Queue the null job, but do not invoke *uucico*.

*uupoll* is usually run by *cron*(5) or by a user who wants to hurry a job along. A typical entry in *crontab* could be:

```
0      0,8,16 * * * /usr/bin/uupoll ihnp4
0      4,12,20 * * * /usr/bin/uupoll convex
```

This polls *ihnp4* at midnight, 0800, and 1600, and *convex* at 0400, noon, and 2000.

If the local machine is already running *uucico* every hour and has a limited number of outgoing modems, a more elegant approach might be:

```
0      0,8,16 * * * /usr/bin/uupoll -n ihnp4
0      4,12,20 * * * /usr/bin/uupoll -n convex
5      * * * * * /usr/lib/uucp/uucico -r1
```

This queues null jobs for the remote sites at the top of hour; they are processed by *uucico* when it runs five minutes later.

## FILES

/usr/lib/uucp/ UUCP internal files/utilities  
/usr/spool/uucp/ Spool directory

## SEE ALSO

uucp(1C), uux(1C), uucico(8C)

**NAME**

uusnap – show snapshot of the UUCP system

**SYNOPSIS**

**uusnap**

**DESCRIPTION**

*uusnap* displays in tabular format a synopsis of the current UUCP situation. The format of each line is as follows:

*site N Cmds N Data N Xqts Message*

Where “site” is the name of the site with work, “N” is a count of each of the three possible types of work (command, data, or remote execute), and “Message” is the current status message for that site as found in the STST file.

Included in “Message” may be the time left before UUCP can re-try the call and the count of the number of times that UUCP has tried (unsuccessfully) to reach the site.

**SEE ALSO**

uucp(1C), uux(1C), uuq(1C), uucico(8C)  
“*UUCP Implementation Guide*”

**NAME**

uuxqt - UUCP execution file interpreter

**SYNOPSIS**

`/usr/lib/uucp/uuxqt [ -xdebug ]`

**DESCRIPTION**

*uuxqt* interprets *execution files* created on a remote system via *uuz*(1C) and transferred to the local system via *uucico*(8C). When a user uses *uuz* to request remote command execution, it is *uuxqt* that actually executes the command. Normally, *uuxqt* is forked from *uucico* to process queued execution files; for debugging, it may also be run manually by the UUCP administrator.

*uuxqt* runs in its own subdirectory, `/usr/spool/uucp/XTMP`. It copies intermediate files to this directory when necessary.

**FILES**

<code>/usr/lib/uucp/L.cmds</code>	Remote command permissions list
<code>/usr/lib/uucp/USERFILE</code>	Remote directory tree permissions list
<code>/usr/spool/uucp/LOGFILE</code>	UUCP system activity log
<code>/usr/spool/uucp/LCK/LCK.XQT</code>	<i>uuxqt</i> lock file
<code>/usr/spool/uucp/X./</code>	Incoming execution file directory
<code>/usr/spool/uucp/XTMP</code>	<i>uuxqt</i> running directory

**SEE ALSO**

*uucp*(1C), *uuz*(1C), *L.cmds*(5), *USERFILE*(5), *uucico*(8C)

**NAME**

verify - verify characteristics of system files

**SYNOPSIS**

*verify* [ *-d directory* ] [ *-i* [*-y*] ] *database* ...

**DESCRIPTION**

*Verify* checks a set of system files to ensure that they have the characteristics specified in a database file. If the *database* filename is neither an absolute pathname nor a relative path that begins with *./* or *../*, then the file is assumed to be in the */usr/lib/verify* directory. The *database* file lists the files to be verified and the characteristics to verify for each file. Characteristics that may be checked are file type, owner, group, and mode. In addition, for symbolic links the "pointed to" file name may be checked.

In its normal mode of operation, *verify* produces a list of differences found. Each characteristic of a file that differs from what was specified in the database is noted in the output. With the *-i* option, the owner, group, and mode may be changed interactively to match the values given in the database.

*verify* is a tool to aid in the diagnosing of system problems and is intended to be used by root. Supplied with each release are a set of database files. These files are located in the directory */usr/lib/verify* and cover virtually all of the standard system files. Each database file contains entries for the files that make up an optional product. Look in */usr/lib/verify* to see what database files are available. The format of the database files is described in *verify(5)*.

**OPTIONS**

- d directory*  
Specify a base directory to prepend to all directory and file names.
- i*  
Run in interactive mode. Every time the owner, group, or mode of a file is found to be incorrect, a prompt is issued offering to correct it. The expected response is either *y* for yes, or *n* for no. A default may also be given as part of the prompt.
- y*  
A YES answer is assumed in response to all questions posed in the interactive mode. That is, all differences noted are automatically corrected.

**DIAGNOSTICS**

Each time an attempt is made to change the owner, group, or mode of a file, the attempt is recorded using *syslogd(8)*. Successful attempts are logged with level LOG\_NOTICE and unsuccessful ones are logged with level LOG\_WARNING.

*verify* exits with a *sysexit* code. Error messages indicate that *verify* is unable to check the current database entry. Warning messages indicate that *verify* uses the default.

**FILES**

*/usr/lib/verify* directory containing standard database files  
*/usr/include/sysexits.h* file containing *sysexit* codes

**SEE ALSO**

*chgrp(1)*, *chmod(1)*, *chown(8)*, *ls(1)*, *syslogd(8)*, *verify(5)*

**NAME**

versatec – versatec filters for use with lpr

**SYNOPSIS**

```
/usr/lib/vpf
/usr/lib/vpfW
/usr/lib/vplotf
/usr/lib/vpltdmp
/usr/lib/vpsf
```

**DESCRIPTION**

The programs *vpf*, *vpfW*, *vplotf*, *vpltdmp*, and *vpsf* are filters invoked by *lpd(8)* to print to a Versatec Printer/Plotter. *Vpf* and *vpfW* are filters for printing alphanumeric data; *vpfW* prints data to a wide plotter. *Vplotf* is a filter for printing *graph(1G)* output. *Vpltdmp* is a filter for printing *plot(1G)* output. *Vpsf* is a filter for printing alphanumeric data with pages appearing side-by-side, for use with a wide plotter. A typical *printcap* entry for a Versatec Plotter which is 2048 pixels wide might be:

```
ve|vers|versatec|Versatec Plotter 2048 pixels wide:\
:lp=/dev/pb4:sd=/usr/spool/ved:af=/usr/adm/veacct:\
:mx#2000:px#2048:py#2200:lf=/usr/spool/vers_log:\
:if=/usr/lib/vpfW:of=/usr/lib/vpfW:\
:gf=/usr/lib/vplotf:vt=/usr/lib/vpltdmp:
```

**FILES**

/usr/tmp/rasterfile containing raster image created by plot(1G)

**SEE ALSO**

graph(1G), lpr(1), plot(1G), plot(3X), spline(1G), vpr(1)

## NAME

*vipw* – edit the password and password restrictions files

## SYNOPSIS

**vipw** [ *editor\_cmds* ]

## DESCRIPTION

*Vipw* simultaneously edits the password file and the password restrictions file. To prevent concurrent updating of the files, locks are used to exclude all but one user from editing the password information. *Vipw* sets the appropriate locks, and does any necessary processing after both files are unlocked. If the files are already being edited with *vipw*, then you will be told to try again later.

To prevent separate editing of the password and password restrictions files, *vipw* combines the information into a single editing session. For each user, *vipw* constructs a single line displaying the union of the user's *pwrestrict* file entry and the user's *passwd* file entry. After editing, *vipw* will separate the information, update both files, and rebuild the password and restrictions databases with *mkpasswd*.

*Vipw* should be the only utility used to update the *pwrestrict* and *passwd* files.

The *vi* editor will be used unless the environment variable EDITOR indicates an alternate editor. Optionally an *editor\_cmds* argument can be given to *vipw*. If supplied, the argument will be passed on to the editor as a command line argument. Assuming *vi* is used as the editor, example uses include:

*vipw +/foo/* will start editing the created password file at the first line containing the text "foo".

*vipw +\%* will start editing at the last line of the created password file.

*Vipw* checks the syntax of all password entries and tries to point out the specific errors made. The errors may be fixed by running *vipw* again. *Vipw* performs a number of consistency checks on the password entry for *root*, and will not allow a password file with a "mangled" root entry to be installed. *Vipw* further restricts root's login shell to one of the shells listed in */etc/shells*. If */etc/shells* does not exist, the C shell */bin/csh*, or the Bourne shell */bin/sh* are the only acceptable shells for root.

## SEE ALSO

*chfn*(1), *chsh*(1), *passwd*(1), *passwd*(5), *pwrestrict*(5), *mkpasswd*(8), *nu*(8)

## FILES

*/etc/ptmp* -- lock file containing new */etc/passwd*

*/etc/rtmp* -- lock file containing new */etc/pwrestrict*

*/etc/vtmp* -- lock file containing both *passwd* and *pwrestrict* fields

## CAVEAT

*Vipw* will complain when a mangled non-root entry has been installed. As mentioned, use *vipw* to re-edit the password files and correct the mistake.

## NAME

`xdump` – fast incremental filesystem dump

## SYNOPSIS

`/etc/xdump key [ argument ... ] filesystem`

## DESCRIPTION

`xdump` copies all files changed after a certain date in the *filesystem* to magnetic tape. The *key* specifies the date and other options about the dump. *key* consists of characters from the set **0123456789fusdWwncblpmtDGIE**.

**0-9** This number is the “dump level”. All files modified since the last date stored in the file `/etc/dumpdates` for the same filesystem at lesser levels will be dumped. If no date is determined by the level, the beginning of time is assumed; thus the option **0** causes the entire filesystem to be dumped.

**f** Place the dump on the next *argument* file instead of the tape. If the name of the file is `-`, `xdump` writes to standard output. The default is `/dev/rmt8`.

**u** If the dump completes successfully, write the date of the beginning of the dump in `/etc/dumpdates`. This file records a separate date for each filesystem and each dump level. The file `/etc/dumpdates` is human readable. It consists of one free-format record per line: filesystem name, increment level, and `ctime(3)` format dump date. If necessary, `/etc/dumpdates` may be edited to change any of the fields.

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

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

**W** `xdump` tells the operator what filesystems need to be dumped. This information is gleaned from the files `/etc/dumpdates` and `/etc/fstab`. The **W** option causes `xdump` to print out, for each filesystem in `/etc/dumpdates`, the most recent dump date and level, and highlights those filesystems that should be dumped. If the **W** option is set, all other options are ignored, and `xdump` exits immediately.

**w** Is like **W**, but prints only those filesystems which need to be dumped.

**n** Whenever `xdump` requires operator attention, notify all operators in the group “operator”, by means similar to a `wall(1)`.

**c** Set size and density defaults for cartridge tape rather than 9-track.

**b** Take the next *argument* to be the blocking factor for tape records. The default is 10.

**l** Take the next *argument* as the read latency in disk sectors. This is the number of sectors the disk will rotate through between reads. The default is 17 sectors.

**p** Take the next *argument* as the amount of data to plan ahead on the reads in units of kilobytes. The default is 600 kilobytes.

**m** Take the next *argument* as the maximum number of fragments to read in a single read request. The default is 64 frags, or 128kbytes, whichever is smaller.

**t** Enable the printing of interesting timing statistics after each phase of the dump.

**D** Take the next *argument* as the minimum number of filesystem fragment sized data buffers to allocate.

**G** Set defaults for a GCR (6250 bpi) dump. This sets the tape unit to be `/dev/rmt16`, the density to 6250, and the blocking factor to 100.

**I** Puts the tape drive into 100 ips streaming mode. The default is 50 ips.

- E** Causes *xdump* to write to tape until finished or until end-of-tape is reached. If end-of-tape is reached, the operator is prompted to change tapes. Even with this option, *xdump* will print an estimate of how many 9-track tapes (not cartridges) will be used for the dump.

*xdump* requires operator intervention on these conditions: end-of-tape, end-of-dump, tape-write error, tape-open error, or disk-read error. In addition to alerting all operators implied by the **n** key, *xdump* interacts with the operator on *xdump*'s control terminal at times when *xdump* can no longer proceed, or if something is seriously wrong. All questions *xdump* poses **must** be answered by typing "yes" or "no".

*xdump* tells the operator what is going on at periodic intervals, including estimates of the number of blocks to be written, the number of tapes it will take, the time to completion, and the time to the tape change.

For each filesystem at your site, establish a sequence of full and incremental dumps on a weekly cycle. Because backing up and restoring a filesystem is a time-consuming process, the sequence you select has an impact on the amount of time required to perform backups and to restore files from dump tapes.

One possible backup schedule for performing dumps follows:

Day	Dump Level
Monday	level 0: backs up all files in the filesystem (full dump)
Tuesday	level 5: backs up all files changed since Monday's full dump
Wednesday	level 5: backs up all files changed since Monday's full dump
Thursday	level 5: backs up all files changed since Monday's full dump
Friday	level 5: backs up all files changed since Monday's full dump

With this method, restoring files requires only two sets of dump tapes: tapes for the full dump and tapes for the most recent incremental dump.

#### CAUTION

*xdump* DIFFERS IN ONE VERY IMPORTANT ASPECT FROM THE ORIGINAL. This version does NOT re-open every file just before it is dumped. If you dump an active file system using this dump, you run a slightly greater risk of dumping blocks which no longer resemble the file to which they used to belong. Guarantees of consistency are valid only for quiescent filesystems.

#### FILES

/dev/rmt8	default tape unit to dump to
/etc/ddate	old format dump date record (obsolete after <b>-J</b> option)
/etc/dumpdates	new format dump date record
/etc/fstab	dump table: filesystems and frequency
/etc/group	to find group <i>operator</i>

#### SEE ALSO

mtio(4), dump(5), fstab(5), restore(8)

#### DIAGNOSTICS

Many, and verbose. Unlike most other utilities, *xdump* returns a 1 upon normal completion.

#### BUGS

Sizes are based on 1600 bpi blocked tape; the raw magtape device has to be used to approach these densities.

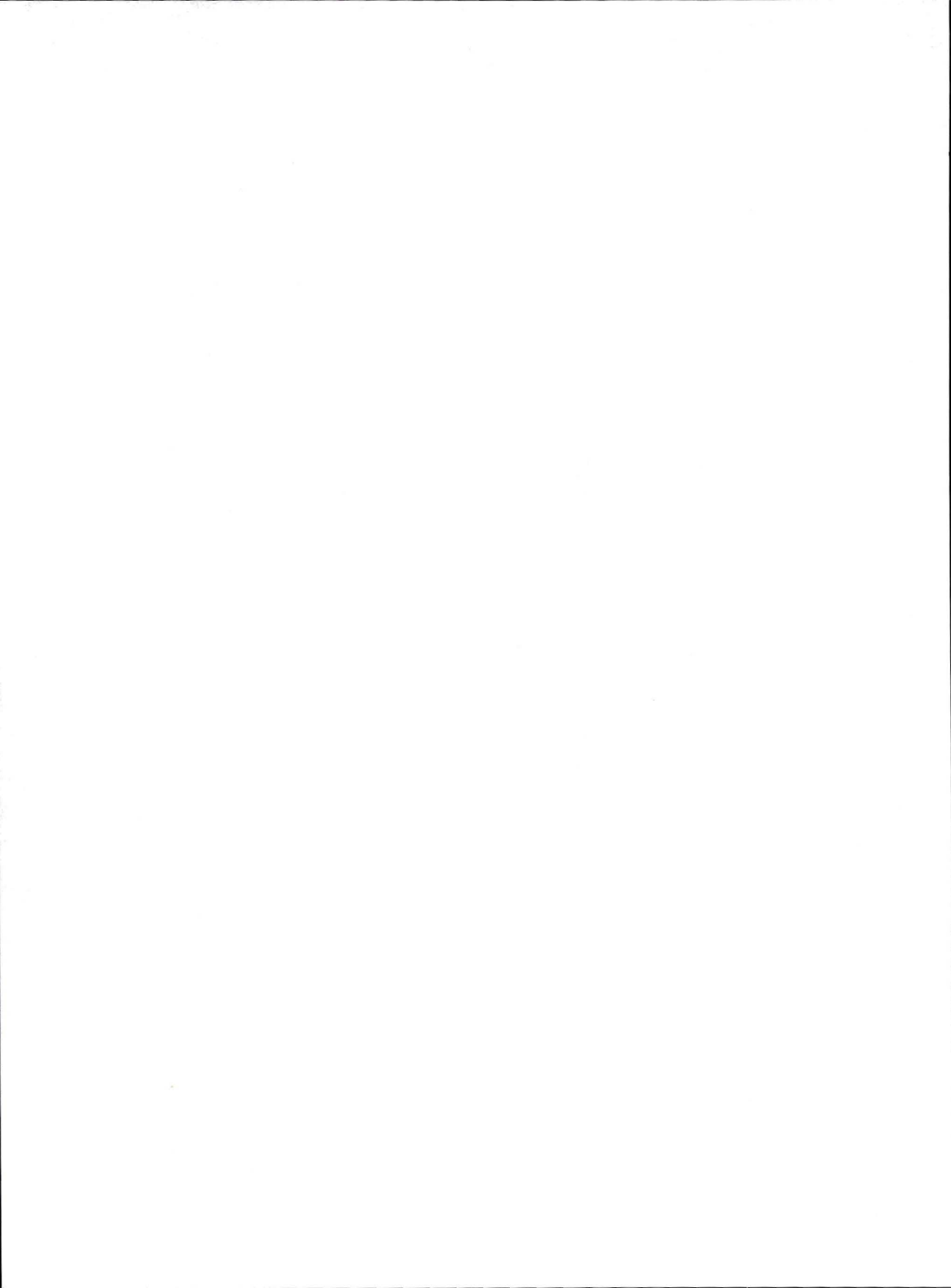
*xdump* does not know about the dump sequence, does not keep track of the tapes scribbled on, does not tell the operator which tape to mount when, and does not provide assistance for the operator running *restore*.

If a different tape blocking factor is used, it must also be used with *restore*.



# Index

---



Description	Product	Man Page
3-way differential file comparison	ConvexOS	DIFF3(1)
680x0 link editor	ConvexOS	LD68(1)
680x0 peephole optimizer	ConvexOS	O68(1)
<i>_exit</i> — terminate a process	ConvexOS	EXIT(2)
<i>_tolower</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>_toupper</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>a.out</i> — CONVEX assembler and link editor output	ConvexOS	A.OUT(5)
<i>a20</i> — cross-assemblers for the 680x0 instruction sets	ConvexOS	A68(1)
<i>a68</i> — cross-assemblers for the 680x0 instruction sets	ConvexOS	A68(1)
<i>abort</i> — generate a fault	ConvexOS	ABORT(3)
<i>abs</i> — standard integral functions	ConvexOS	ABS(3)
absolute value, floor, ceiling functions	ConvexOS	FLOOR(3M)
<i>ac</i> — login accounting	ConvexOS	AC(8)
<i>accept</i> — accept a connection on a socket	ConvexOS	ACCEPT(2)
accept a connection on a socket	ConvexOS	ACCEPT(2)
accept mail for a notesfile	ConvexOS	NFMAIL(8)
<i>access</i> — determine accessibility of file	ConvexOS	ACCESS(2)
<i>accounting</i> — periodic accounting utilities	ConvexOS	ACCOUNTING(8)
accounting file conversion	ConvexOS	ACCTCONV(8)
accounting summary awk scripts	ConvexOS	SUMSCRIPTS(8)
<i>acct</i> — execution accounting file	ConvexOS	ACCT(5)
<i>acct</i> — log generated by bill command	ConvexOS	BILL-ACCT(5)
<i>acct</i> — printer accounting file	ConvexOS	LPD-ACCT(5)
<i>acct</i> — tape allocation accounting file	ConvexOS	TP-ACCT(5)
<i>acct</i> — turn accounting on or off	ConvexOS	ACCT(2)
<i>acctconv</i> — accounting file conversion	ConvexOS	ACCTCONV(8)
<i>accton</i> — system accounting	ConvexOS	SA(8)
<i>acos</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>acosf</i> — trigonometric functions	ConvexOS	SIN(3M)
acquire and display HYPERchannel adapter statistics	ConvexOS	HYSTAT(8C)
<i>activities</i> — activity list	ConvexOS	ACTIVITIES(5)
activity list	ConvexOS	ACTIVITIES(5)
<i>actwho</i> — group-activity access control file	ConvexOS	ACTWHO(5)
<i>adb</i> — debugger	ConvexOS	ADB(1)
<i>adb68</i> — debugger	ConvexOS	ADB68(1)
<i>adbccu</i> — debugger	ConvexOS	ADBCCU(8)
add a new user to the system	ConvexOS	NU(8)
add a swap device for interleaved paging/swapping	ConvexOS	SWAPON(2)
add access rights to a set of Notesfiles	ConvexOS	NFACCESS(8)
<i>admntent</i> — get file system descriptor file entry	ConvexOS	GETMNTENT(3)
address resolution display and control	ConvexOS	ARP(8C)
Address Resolution Protocol	ConvexOS	ARP(4P)
<i>adjtime</i> — adjust time	ConvexOS	ADJTIME(2)
adjust time	ConvexOS	ADJTIME(2)
advisory record locking on files	ConvexOS	LOCKF(3)
<i>alarm</i> — schedule signal after specified time	ConvexOS	ALARM(3C)
<i>aliases</i> — aliases file for sendmail	ConvexOS	ALIASES(5)
aliases file for sendmail	ConvexOS	ALIASES(5)
aligned memory allocator	ConvexOS	VALLOC(3)

Description	Product	Man Page
<i>alloca</i> — memory allocator	ConvexOS	MALLOC(3)
<i>alphasort</i> — scan a directory	ConvexOS	SCANDIR(3)
alter priority of running processes	ConvexOS	RENICE(8)
analyze and disperse compiler error messages	ConvexOS	ERROR(1)
analyze surface characteristics of a document	ConvexOS	STYLE(1)
ANSI-labeled-tape processing filter/daemon	ConvexOS	ANSIDAEMON(8)
<i>ansidaemon</i> — ANSI-labeled-tape processing filter/daemon	ConvexOS	ANSIDAEMON(8)
<i>ansitar</i> — read or write ANSI multifile labeled tapes	ConvexOS	ANSITAR(1)
<i>apply</i> — apply a command to a set of arguments	ConvexOS	APPLY(1)
apply a command to a set of arguments	ConvexOS	APPLY(1)
apply or remove an advisory lock on an open file	ConvexOS	FLOCK(2)
<i>apropos</i> — display on-line reference manual information	ConvexOS	MAN(1)
<i>apropos</i> — locate commands by keyword lookup	ConvexOS	APROPOS(1)
<i>ar</i> — archive (library) file format	ConvexOS	AR(5)
<i>ar</i> — archive and library maintainer	ConvexOS	AR(1)
arbitrary-precision arithmetic language	ConvexOS	BC(1)
<i>arc</i> — graphics interface	ConvexOS	PLOT(3X)
archive (library) file format	ConvexOS	AR(5)
archive and library maintainer	ConvexOS	AR(1)
archive notesfiles	ConvexOS	NFARCHIVE(8)
<i>arp</i> — address resolution display and control	ConvexOS	ARP(8C)
<i>arp</i> — Address Resolution Protocol	ConvexOS	ARP(4P)
<i>as</i> — assembler for the CONVEX supercomputer instruction set	ConvexOS	AS(1)
<i>as88</i> — Assembly cross-compiler to generate MC88100 code on a CONVEX C Series Computer.	ConvexOS	AS88(1)
<i>ascii</i> — map of ASCII character set	ConvexOS	ASCII(7)
<i>asctime</i> — date and time manipulation routines	ConvexOS	CTIME(3)
<i>asin</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>asinf</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>asiostat</i> — wait, then return asynchronous I/O byte count	ConvexOS	ASIOSTAT(2)
assembler for the CONVEX supercomputer instruction set	ConvexOS	AS(1)
Assembly cross-compiler to generate MC88100 code on a CONVEX C Series Computer.	ConvexOS	AS88(1)
<i>assert</i> — program verification	ConvexOS	ASSERT(3X)
assign buffering to a stream	ConvexOS	SETBUF(3S)
<i>at</i> — execute commands at a later time	ConvexOS	AT(1)
<i>atan</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>atan2</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>atan2f</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>atanf</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>atexit</i> — terminate a process after flushing any pending output	ConvexOS	EXIT(3)
<i>atof</i> — convert ASCII to numbers	ConvexOS	ATOF(3)
<i>atoi</i> — convert ASCII to numbers	ConvexOS	ATOF(3)

Description	Product	Man Page
<i>atol</i> — convert ASCII to numbers	ConvexOS	ATOF(3)
<i>atoll</i> — convert ASCII to numbers	ConvexOS	ATOF(3)
atomically release blocked signals and wait for interrupt	ConvexOS	SIGPAUSE(2)
<i>atq</i> — print the queue of jobs waiting to be run	ConvexOS	ATQ(1)
<i>atrm</i> — remove jobs spooled by at	ConvexOS	ATRM(1)
<i>atrun</i> — execute commands at a later time	ConvexOS	AT(1)
attach serial lines as network interfaces	ConvexOS	SLATTACH(8C)
<i>autoconf</i> — diagnostics from the autoconfiguration code	ConvexOS	AUTOCONF(4)
<i>autoseq</i> — news system	ConvexOS	NOTES(1)
<i>avail</i> — maintain and process system availability information	ConvexOS	AVAIL(8)
await completion of process	ConvexOS	WAIT(1)
<i>awk</i> — pattern scanning and processing language	ConvexOS	AWK(1)
<i>b.out</i> — executable file format	ConvexOS	B.OUT(5)
<i>backup</i> — statd directory and file structures	ConvexOS	STATMON(5)
<i>badlogins</i> — log of failed login attempts	ConvexOS	BADLOGINS(5)
<i>basename</i> — strip filename affixes	ConvexOS	BASENAME(1)
<i>bc</i> — arbitrary-precision arithmetic language	ConvexOS	BC(1)
<i>bcmp</i> — bit and byte string operations	ConvexOS	BSTRING(3)
<i>bcopy</i> — bit and byte string operations	ConvexOS	BSTRING(3)
be notified if mail arrives and who it is from	ConvexOS	BIFF(1)
be repetitively affirmative	ConvexOS	YES(1)
bessel functions	ConvexOS	J0(3M)
better random number generator; routines for changing generators	ConvexOS	RANDOM(3)
<i>biff</i> — be notified if mail arrives and who it is from	ConvexOS	BIFF(1)
biff server	ConvexOS	COMSAT(8C)
<i>bill</i> — change current billing account	ConvexOS	BILL(1)
<i>bind</i> — bind a name to a socket	ConvexOS	BIND(2)
bind a name to a socket	ConvexOS	BIND(2)
bind a socket to a privileged IP port	ConvexOS	BINDRESVPORT(3N)
<i>bindresvport</i> — bind a socket to a privileged IP port	ConvexOS	BINDRESVPORT(3N)
<i>binmail</i> — send or receive mail among users	ConvexOS	BINMAIL(1)
bit and byte string operations	ConvexOS	BSTRING(3)
block or unblock signals	ConvexOS	SIGBLOCK(2)
block or unblock signals	ConvexOS	SIGPROCMASK(3)
<i>brk</i> — change data segment size	ConvexOS	BRK(2)
<i>bsearch</i> — quicker sort and search	ConvexOS	QSORT(3)
buffered binary input/output	ConvexOS	FREAD(3S)
build RCS file from SCCS file	ConvexOS	SCCSTORCS(1)
build special file	ConvexOS	MKNOD(8)
<i>bzero</i> — bit and byte string operations	ConvexOS	BSTRING(3)
C cross-compiler to generate MC88100 code on a CONVEX C Series Computer.	ConvexOS	CC88(1)
C cross-compilers for use on the CONVEX Computer	ConvexOS	CC68(1)
C preprocessor	ConvexOS	CPP(1)
C program verifier	ConvexOS	LINT(1)

Description	Product	Man Page
C program verifier	ConvexOS	LINT68(1)
<i>ca</i> — terminal multiplexor	ConvexOS	CA(4)
<i>cabs</i> — Euclidean distance	ConvexOS	HYPOT(3M)
<i>cal</i> — print calendar	ConvexOS	CAL(1)
<i>calendar</i> — reminder service	ConvexOS	CALENDAR(1)
<i>calloc</i> — memory allocator	ConvexOS	MALLOC(3)
cartridge tape archiver	ConvexOS	CTAR(8)
cartridge tape driver	ConvexOS	TC(4)
cartridge tape interface	ConvexOS	CT(4)
<i>cat</i> — catenate and print	ConvexOS	CAT(1)
catenate and print	ConvexOS	CAT(1)
<i>catinfo</i> — create the formatted files for the info system	ConvexOS	CATINFO(8)
<i>catman</i> — create the cat files for the manual	ConvexOS	CATMAN(8)
<i>cb</i> — COMBOARD communications interface	ConvexOS	CB(4)
<i>cc</i> — CONVEX C compiler	ConvexOS	PCC(1)
<i>cc20</i> — C cross-compilers for use on the CONVEX Computer	ConvexOS	CC68(1)
<i>cc68</i> — C cross-compilers for use on the CONVEX Computer	ConvexOS	CC68(1)
<i>cc88</i> — C cross-compiler to generate MC88100 code on a CONVEX C Series Computer.	ConvexOS	CC88(1)
<i>ccat</i> — compress and uncompress files, and cat them	ConvexOS	COMPACT(1)
<i>ccrypt</i> — CONVEX encode/decode	ConvexOS	CCRYPT(1)
<i>ccu</i> — channel control unit pseudo-device	ConvexOS	CCU(4)
<i>cd</i> — change working directory	ConvexOS	CD(1)
<i>cdecl</i> — Compose C declarations	ConvexOS	CDECL(1)
<i>cdown</i> — controller download utility	ConvexOS	CDOWN(1)
<i>cdump</i> — controller crashdump utility	ConvexOS	CDUMP(1)
<i>ceil</i> — absolute value, floor, ceiling functions	ConvexOS	FLOOR(3M)
<i>cfgetispeed</i> — get/set terminal input/output speed	ConvexOS	CFGETOSPEED(3)
<i>cfgetospeed</i> — get/set terminal input/output speed	ConvexOS	CFGETOSPEED(3)
<i>cfree</i> — memory allocator	ConvexOS	MALLOC(3)
<i>cfsetispeed</i> — get/set terminal input/output speed	ConvexOS	CFGETOSPEED(3)
<i>cfsetospeed</i> — get/set terminal input/output speed	ConvexOS	CFGETOSPEED(3)
<i>chall</i> — change mode, owner and/or group of a file	ConvexOS	CHALL(8)
change attributes of a memory region in a process' address space	ConvexOS	MREMAP(2)
change attributes used for future labeled tape files	ConvexOS	TPATTR(1)
change charges for share system usage calculations	ConvexOS	CHARGE(8)
change current billing account	ConvexOS	BILL(1)
change current working directory	ConvexOS	CHDIR(2)
change data segment size	ConvexOS	BRK(2)
change default login shell	ConvexOS	CHSH(1)
change finger entry	ConvexOS	CHFN(1)
change group	ConvexOS	CHGRP(1)
change login password	ConvexOS	PASSWD(1)
change magic number of executable file	ConvexOS	CHMAGIC(1)
change mode	ConvexOS	CHMOD(1)
change mode of file	ConvexOS	CHMOD(2)

Description	Product	Man Page
change mode of file	ConvexOS	FCHMOD(2)
change mode, owner and/or group of a file	ConvexOS	CHALL(8)
change name of shares file	ConvexOS	SHARESFILE(3)
change owner	ConvexOS	CHOWN(8)
change owner and group of a file	ConvexOS	CHOWN(2)
change owner and group of file by descriptor	ConvexOS	FCHOWN(2)
change RCS file attributes	ConvexOS	RCS(1)
change root directory	ConvexOS	CHROOT(2)
change share data base information for one or more users	ConvexOS	LIM(8)
change the name of a file	ConvexOS	RENAME(2)
change working directory	ConvexOS	CD(1)
channel control unit pseudo-device	ConvexOS	CCU(4)
character testing and mapping macros	ConvexOS	CTYPE(3)
<i>charge</i> — change charges for share system usage calculations	ConvexOS	CHARGE(8)
<i>charges</i> — show charges for share system usage calculations	ConvexOS	CHARGES(1)
<i>chdir</i> — change current working directory	ConvexOS	CHDIR(2)
check file system quota consistency	ConvexOS	QUOTACHECK(8)
check for new notesfile articles	ConvexOS	CHECKNOTES(1)
check in RCS revisions	ConvexOS	CI(1)
check out RCS revisions	ConvexOS	CO(1)
check nroff/troff files	ConvexOS	CHECKNR(1)
<i>checkeq</i> — typeset mathematics	ConvexOS	NEQN(1)
<i>checknotes</i> — check for new notesfile articles	ConvexOS	CHECKNOTES(1)
<i>checknr</i> — check nroff/troff files	ConvexOS	CHECKNR(1)
checkpoint a process or process family	ConvexOS	CHKPNT(1)
checkpoint a process or process family	ConvexOS	CHKPNT(3)
<i>chfn</i> — change finger entry	ConvexOS	CHFN(1)
<i>chgrp</i> — change group	ConvexOS	CHGRP(1)
<i>chkpnt</i> — checkpoint a process or process family	ConvexOS	CHKPNT(1)
<i>chkpnt</i> — checkpoint a process or process family	ConvexOS	CHKPNT(3)
<i>chkpnt</i> — process checkpoint file format	ConvexOS	CHKPNT(5)
<i>chmagic</i> — change magic number of executable file	ConvexOS	CHMAGIC(1)
<i>chmod</i> — change mode	ConvexOS	CHMOD(1)
<i>chmod</i> — change mode of file	ConvexOS	CHMOD(2)
<i>chown</i> — change owner	ConvexOS	CHOWN(8)
<i>chown</i> — change owner and group of a file	ConvexOS	CHOWN(2)
<i>chroot</i> — change root directory	ConvexOS	CHROOT(2)
<i>chsh</i> — change default login shell	ConvexOS	CHSH(1)
<i>ci</i> — check in RCS revisions	ConvexOS	CI(1)
<i>circle</i> — graphics interface	ConvexOS	PLOT(3X)
clean up defunct user inode structures	ConvexOS	SHARER(8)
<i>clear</i> — clear terminal screen	ConvexOS	CLEAR(1)
clear i-node	ConvexOS	CLRI(8)
clear terminal screen	ConvexOS	CLEAR(1)
<i>clearerr</i> — stream status inquiries	ConvexOS	FERROR(3S)
<i>clock</i> — get processor time used	ConvexOS	CLOCK(3)

Description	Product	Man Page
<i>close</i> — delete a descriptor	ConvexOS	CLOSE(2)
close down the system at a given time	ConvexOS	SHUTDOWN(8)
close or flush a stream	ConvexOS	FCLOSE(3S)
close share data base file	ConvexOS	CLOSESHARES(3)
<i>closedir</i> — directory operations	ConvexOS	DIRECTORY(3)
<i>closelog</i> — control system log	ConvexOS	SYSLOG(3)
<i>closepl</i> — graphics interface	ConvexOS	PLOT(3X)
<i>closeshares</i> — close share data base file	ConvexOS	CLOSESHARES(3)
<i>clri</i> — clear i-node	ConvexOS	CLRI(8)
<i>cmp</i> — compare two files	ConvexOS	CMP(1)
<i>co</i> — check out RCS revisions	ConvexOS	CO(1)
<i>col</i> — filter reverse line feeds	ConvexOS	COL(1)
<i>colert</i> — filter nroff output for CRT previewing	ConvexOS	COLERT(1)
<i>colrm</i> — remove columns from a file	ConvexOS	COLRM(1)
COMBOARD communications interface	ConvexOS	CB(4)
<i>comm</i> — select or reject lines common to two sorted files	ConvexOS	COMM(1)
command language	ConvexOS	SH(1)
command script for auto-reboot and daemons	ConvexOS	RC(8)
<i>commcap</i> — HASP Host communications capability database	ConvexOS	COMMCAP(5)
<i>compact</i> — compress and uncompress files, and cat them	ConvexOS	COMPACT(1)
compact list of users who are on the system	ConvexOS	USERS(1)
compare RCS revisions	ConvexOS	RCSDIFF(1)
compare two files	ConvexOS	CMP(1)
Compose C declarations	ConvexOS	CDECL(1)
compress and uncompress files, and cat them	ConvexOS	COMPACT(1)
compute connect time from wtmp and bill-act	ConvexOS	CONNECTTIME(8)
compute the difference between two times.	ConvexOS	DIFFTIME(3)
compute user and group disk-space totals	ConvexOS	DISKUSE(8)
computer aided instruction about UNIX	ConvexOS	LEARN(1)
<i>comsat</i> — biff server	ConvexOS	COMSAT(8C)
condition command	ConvexOS	TEST(1)
configurable pathname variables	ConvexOS	PATHCONF(3)
configure and display a Share configuration	ConvexOS	SHARECF(8)
configure network interface parameters	ConvexOS	IFCONFIG(8c)
configure the tape system	ConvexOS	TPCONFIG(8)
<i>connect</i> — initiate a connection on a socket	ConvexOS	CONNECT(2)
<i>connecttime</i> — compute connect time from wtmp and bill-act	ConvexOS	CONNECTTIME(8)
<i>cons</i> — console interface	ConvexOS	CONS(4)
console interface	ConvexOS	CONS(4)
construct a file system	ConvexOS	MKFS(8)
construct a new file system	ConvexOS	NEWFS(8)
construct Makefile dependency list	ConvexOS	MKDEP(1)
<i>cont</i> — graphics interface	ConvexOS	PLOT(3X)
<i>contact</i> — submit a CONVEX problem report	ConvexOS	CONTACT(1)
<i>contactcap</i> — system configuration file for contact	ConvexOS	CONTACTCAP(5)

Description	Product	Man Page
continually watch the end of a file	ConvexOS	SNIFF(1)
control daemons	ConvexOS	DMON_IOCTL(2)
control device	ConvexOS	_IOCTL(2)
control maximum system resource consumption	ConvexOS	GETRLIMIT(2)
control maximum system resource consumption	ConvexOS	VLIMIT(3C)
control system log	ConvexOS	SYSLOG(3)
controller crashdump utility	ConvexOS	CDUMP(1)
controller download utility	ConvexOS	CDOWN(1)
controller reset utility	ConvexOS	CRESET(1)
conventional names for terminals	ConvexOS	TERM(7)
conversion program	ConvexOS	UNITS(1)
convert and copy a file	ConvexOS	DD(1)
convert archives to random libraries	ConvexOS	RANLIB(1)
convert ASCII to numbers	ConvexOS	ATOF(3)
convert NIC standard format host tables	ConvexOS	HTABLE(8C)
convert values between host and network byte order	ConvexOS	BYTEORDER(3N)
CONVEX assembler and link editor output	ConvexOS	A.OUT(5)
CONVEX C compiler	ConvexOS	PCC(1)
CONVEX encode/decode	ConvexOS	CCRYPT(1)
copy	ConvexOS	CP(1)
copy directory	ConvexOS	CPALL(1)
copy file archives in and out	ConvexOS	CPIO(1)
<i>core</i> — format of memory image file	ConvexOS	CORE(5)
<i>cos</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>cosf</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>cosh</i> — hyperbolic functions	ConvexOS	SINH(3M)
<i>coshf</i> — hyperbolic functions	ConvexOS	SINH(3M)
<i>cp</i> — copy	ConvexOS	CP(1)
<i>cpall</i> — copy directory	ConvexOS	CPALL(1)
<i>cpio</i> — copy file archives in and out	ConvexOS	CPIO(1)
<i>cpio</i> — POSIX compatible extended cpio archive file format	ConvexOS	CPIO(5)
<i>cpp</i> — C preprocessor	ConvexOS	CPP(1)
<i>cpp88</i> — MC88100 C Preprocessor	ConvexOS	CPP88(1)
<i>cpr</i> — print "C" files	ConvexOS	CPR(1)
CPU Configuration	ConvexOS	CPUCONF(8)
<i>cpuconf</i> — CPU Configuration	ConvexOS	CPUCONF(8)
crash dump tape	ConvexOS	CRASHDUMP(5)
<i>crashdump</i> — crash dump tape	ConvexOS	CRASHDUMP(5)
<i>crashdump</i> — write a crash dump to tape	ConvexOS	CRASHDUMP(8)
<i>crashread</i> — read physical memory from a crash dump tape	ConvexOS	CRASHREAD(8)
<i>creat</i> — create a new file	ConvexOS	CREATE(2)
create a new file	ConvexOS	CREATE(2)
create a new labeled tape	ConvexOS	TPLABEL(1)
create a new process	ConvexOS	FORK(2)
create a new thread	ConvexOS	THREAD_CREATE(2)
create a pair of connected sockets	ConvexOS	SOCKETPAIR(2)
create a tags file	ConvexOS	CTAGS(1)

Description	Product	Man Page
create a temporary file or generate a unique file name.	ConvexOS	TMPFILE(3S)
create an endpoint for communication	ConvexOS	SOCKET(2)
create an error message file by massaging C source	ConvexOS	MKSTR(1)
create an interprocess communication channel	ConvexOS	PIPE(2)
create and delete notesfiles	ConvexOS	MKNF(8)
create session and set process group ID	ConvexOS	SETSID(2)
create the cat files for the manual	ConvexOS	CATMAN(8)
create the formatted files for the info system	ConvexOS	CATINFO(8)
<i>cref</i> — cross reference program	ConvexOS	CREF(1)
<i>reset</i> — controller reset utility	ConvexOS	RESET(1)
<i>cron</i> — user clock daemon	ConvexOS	CRON(1)
cron command list file	ConvexOS	CRONTAB(5)
<i>crontab</i> — cron command list file	ConvexOS	CRONTAB(5)
cross reference program	ConvexOS	CREF(1)
cross-assemblers for the 680x0 instruction sets	ConvexOS	A68(1)
<i>crypt</i> — encode/decode	ConvexOS	CRYPT(1)
<i>crypt</i> — encryption operations	ConvexOS	CRYPT(3)
<i>cs68</i> — generate checksum for 680x0 b.out format file	ConvexOS	CS68(1)
<i>cs</i> — shell (command interpreter) with C-like syntax	ConvexOS	CSH(1)
<i>ct</i> — cartridge tape interface	ConvexOS	CT(4)
<i>ctags</i> — create a tags file	ConvexOS	CTAGS(1)
<i>ctar</i> — cartridge tape archiver	ConvexOS	CTAR(8)
<i>ctermid</i> — generate terminal pathname	ConvexOS	CTERMID(3)
<i>ctime</i> — date and time manipulation routines	ConvexOS	CTIME(3)
<i>current</i> — statd directory and file structures	ConvexOS	STATMON(5)
<i>current_rmts</i> — HASP Host communications boards database	ConvexOS	CURRENT_RMTS(5)
<i>curves</i> — screen functions with “optimal” cursor motion	ConvexOS	CURSES(3X)
<i>cuserid</i> — get user name	ConvexOS	CUSERID(3)
<i>cvzftruncate</i> — truncate arbitrary blocks of a file.	ConvexOS	CVXTRUNCATE(2)
<i>cvzfstat</i> — get file status	ConvexOS	CVXSTAT(2)
<i>cvzftruncate</i> — truncate arbitrary blocks of a file.	ConvexOS	CVXTRUNCATE(2)
<i>cvzfstat</i> — get file status	ConvexOS	CVXSTAT(2)
<i>cvzprusage</i> — get information about parallel resource utilization	ConvexOS	CVXPRUSAGE(2)
<i>cvzstat</i> — get file status	ConvexOS	CVXSTAT(2)
<i>cvztruncate</i> — truncate arbitrary blocks of a file.	ConvexOS	CVXTRUNCATE(2)
<i>da</i> — mass storage disk interface	ConvexOS	DA(4)
daemon interface to kernel RPC facility	ConvexOS	KRPC(2)
Daemon operations on files.	ConvexOS	DMON_FCNTL(2)
DARPA Internet File Transfer Protocol server	ConvexOS	FTPD(8)
DARPA Internet user name directory service	ConvexOS	WHOIS(1C)
DARPA TELNET protocol server	ConvexOS	TELNETD(8C)
data base of magic number checks used by the file(1) utility	ConvexOS	MAGIC(5)
data base of terminal types by port	ConvexOS	TTYTYPE(5)
data base subroutines	ConvexOS	DBM(3X)
data base subroutines	ConvexOS	NDBM(3)

Description	Product	Man Page
data sink	ConvexOS	NULL(4)
<i>date</i> — print and set the date	ConvexOS	DATE(1)
date and time manipulation routines	ConvexOS	CTIME(3)
<i>dbadd</i> — EMACS database manipulation	ConvexOS	DBADD(1)
<i>dbcreate</i> — EMACS database manipulation	ConvexOS	DBADD(1)
<i>dblist</i> — EMACS database manipulation	ConvexOS	DBADD(1)
<i>dbm_clearerr</i> — data base subroutines	ConvexOS	NDBM(3)
<i>dbm_close</i> — data base subroutines	ConvexOS	NDBM(3)
<i>dbm_delete</i> — data base subroutines	ConvexOS	NDBM(3)
<i>dbm_error</i> — data base subroutines	ConvexOS	NDBM(3)
<i>dbm_fetch</i> — data base subroutines	ConvexOS	NDBM(3)
<i>dbm_firstkey</i> — data base subroutines	ConvexOS	NDBM(3)
<i>dbm_nextkey</i> — data base subroutines	ConvexOS	NDBM(3)
<i>dbm_open</i> — data base subroutines	ConvexOS	NDBM(3)
<i>dbm_store</i> — data base subroutines	ConvexOS	NDBM(3)
<i>dbm_init</i> — data base subroutines	ConvexOS	DBM(3X)
<i>dbprint</i> — EMACS database manipulation	ConvexOS	DBADD(1)
<i>dc</i> — desk calculator	ConvexOS	DC(1)
<i>dcheck</i> — file system directory consistency check	ConvexOS	DCHECK(8)
<i>devtir</i> — IEEE/native floating point mode conversion routines.	ConvexOS	RCVTIR(3M)
<i>dd</i> — convert and copy a file	ConvexOS	DD(1)
<i>dd</i> — VME Dual SMD/ESDI Mass storage disk interface	ConvexOS	DD(4)
debugger	ConvexOS	ADB(1)
debugger	ConvexOS	ADBCCU(8)
debugger	ConvexOS	ADB68(1)
encode/decode	ConvexOS	CRYPT(1)
define the standard user shells	ConvexOS	SHELLS(5)
<i>delete</i> — data base subroutines	ConvexOS	DBM(3X)
delete a descriptor	ConvexOS	CLOSE(2)
<i>deroff</i> — remove nroff, troff, tbl and eqn constructs	ConvexOS	DEROFF(1)
describe what a command is	ConvexOS	WHATIS(1)
desk calculator	ConvexOS	DC(1)
determine accessibility of file	ConvexOS	ACCESS(2)
determine file type	ConvexOS	FILE(1)
<i>devices</i> — UUCP device description file	ConvexOS	L-DEVICES(5)
<i>df</i> — disk free	ConvexOS	DF(1)
diagnostics from the autoconfiguration code	ConvexOS	AUTOCONF(4)
<i>dialcodes</i> — UUCP phone number index file	ConvexOS	L-DIALCODES(5)
<i>diction</i> — print wordy sentences; thesaurus for diction	ConvexOS	DICTION(1)
<i>diction</i> — print wordy sentences; thesaurus for diction	ConvexOS	EXPLAIN(1)
<i>diff</i> — differential file and directory comparator	ConvexOS	DIFF(1)
<i>diff3</i> — 3-way differential file comparison	ConvexOS	DIFF3(1)
differential file and directory comparator	ConvexOS	DIFF(1)
<i>diffh</i> — differential file and directory comparator	ConvexOS	DIFF(1)
<i>difftime</i> — compute the difference between two times.	ConvexOS	DIFFTIME(3)
<i>dir</i> — format of directories	ConvexOS	DIR(5)
directories to export to NFS clients	ConvexOS	EXPORTS(5)

Description	Product	Man Page
directory operations	ConvexOS	DIRECTORY(3)
<i>dis</i> — display input as refreshed page	ConvexOS	DIS(1)
enable/disable logins on a tty	ConvexOS	ON(8)
disassemble a 680x0 object file	ConvexOS	TEXT68(1)
disk description file	ConvexOS	DISKTAB(5)
disk free	ConvexOS	DF(1)
<i>disktab</i> — disk description file	ConvexOS	DISKTAB(5)
<i>diskuse</i> — compute user and group disk-space totals	ConvexOS	DISKUSE(8)
display disk usage and limits	ConvexOS	QUOTA(1)
display formatted dump of a 680x0 object file	ConvexOS	PR68(1)
display idle time	ConvexOS	IDLE(1)
display input as refreshed page	ConvexOS	DIS(1)
display on-line reference manual information	ConvexOS	MAN(1)
set/display version numbers	ConvexOS	VERS(1)
<i>div</i> — standard integral functions	ConvexOS	ABS(3)
<i>dm</i> — Raster Technologies Model One/80 driver	ConvexOS	DM(4)
<i>dmon_fcntl</i> — Daemon operations on files.	ConvexOS	DMON_FCNTL(2)
<i>dmon_ioctl</i> — control daemons	ConvexOS	DMON_IOCTL(2)
<i>dn_comp</i> — resolver routines	ConvexOS	RESOLVER(3)
<i>dn_expand</i> — resolver routines	ConvexOS	RESOLVER(3)
do underlining	ConvexOS	UL(1)
<i>drum</i> — paging device	ConvexOS	DRUM(4)
<i>du</i> — Integrated Disk Controller IPI-2 Logical Level for Disk special file	ConvexOS	DU(4)
<i>du</i> — summarize disk usage	ConvexOS	DU(1)
<i>dump</i> — incremental dump format	ConvexOS	DUMP(5)
<i>dump</i> — incremental filesystem dump	ConvexOS	DUMP(8)
dump core and log it in a notesfile	ConvexOS	NFABORT(3)
dump file system information	ConvexOS	DUMPFS(8)
dump file systems across the network	ConvexOS	RDUMP(8C)
<i>dumpdates</i> — incremental dump format	ConvexOS	DUMP(5)
<i>dumpfs</i> — dump file system information	ConvexOS	DUMPFS(8)
<i>dup</i> — duplicate a descriptor	ConvexOS	DUP(2)
<i>dup2</i> — duplicate a descriptor	ConvexOS	DUP(2)
duplicate a descriptor	ConvexOS	DUP(2)
<i>e</i> — text editor	ConvexOS	EX(1)
<i>echo</i> — echo arguments	ConvexOS	ECHO(1)
echo arguments	ConvexOS	ECHO(1)
<i>ecvt</i> — output conversion	ConvexOS	ECVT(3)
<i>ed</i> — text editor	ConvexOS	ED(1)
<i>edactwho</i> — edit /etc/actwho file	ConvexOS	EDACTWHO(8)
<i>edata</i> — last locations in program	ConvexOS	END(3)
<i>edit</i> — text editor	ConvexOS	EX(1)
edit /etc/actwho file	ConvexOS	EDACTWHO(8)
edit the password and password restrictions files	ConvexOS	VIPW(8)
edit user quotas	ConvexOS	EDQUOTA(8)
<i>edquota</i> — edit user quotas	ConvexOS	EDQUOTA(8)
<i>egrep</i> — search a file for a pattern	ConvexOS	GREP(1)
eliminate .so's from nroff input	ConvexOS	SOELIM(1)

Description	Product	Man Page
<i>emacs</i> — GNU project Emacs	ConvexOS	EMACS(1)
EMACS database manipulation	ConvexOS	DBADD(1)
enable or disable logging of failed file accesses	ConvexOS	FAILLOG(2)
<i>encrypt</i> — encryption operations	ConvexOS	CRYPT(3)
encryption operations	ConvexOS	CRYPT(3)
<i>end</i> — last locations in program	ConvexOS	END(3)
<i>endactent</i> — get activity file entry	ConvexOS	GETACTENT(3)
<i>endacwent</i> — get actwho file entry	ConvexOS	GETACWENT(3)
<i>endfsent</i> — get file system descriptor file entry	ConvexOS	GETFSENT(3X)
<i>endgrent</i> — get group file entry	ConvexOS	GETGRENT(3)
<i>endhostent</i> — get network host entry	ConvexOS	GETHOSTBYNAME(3N)
<i>endmntent</i> — get file system descriptor file entry	ConvexOS	GETMNTENT(3)
<i>endnetent</i> — get network entry	ConvexOS	GETNETENT(3N)
<i>endprotoent</i> — get protocol entry	ConvexOS	GETPROTOENT(3N)
<i>endpwent</i> — get password file entry	ConvexOS	GETPWENT(3)
<i>endpwrestent</i> — get entry from password restrictions file	ConvexOS	GETPWRESTENT(3)
<i>endservent</i> — get service entry	ConvexOS	GETSERVENT(3N)
<i>endttyent</i> — get ttys file entry	ConvexOS	GETTTYENT(3)
<i>endusershell</i> — get legal user shells	ConvexOS	GETUSERSHELL(3)
<i>environ</i> — execute a file	ConvexOS	EXECL(3)
<i>environ</i> — user environment	ConvexOS	ENVIRON(7)
<i>eqnchar</i> — special character definitions for eqn	ConvexOS	EQNCHAR(7)
<i>erase</i> — graphics interface	ConvexOS	PLOT(3X)
<i>erf</i> — error functions	ConvexOS	ERF(3M)
<i>erfc</i> — error functions	ConvexOS	ERF(3M)
<i>errno</i> — header file relating to error reporting	ConvexOS	ERRNO.H(3)
<i>errno.h</i> — header file relating to error reporting	ConvexOS	ERRNO.H(3)
<i>error</i> — analyze and disperse compiler error messages	ConvexOS	ERROR(1)
error functions	ConvexOS	ERF(3M)
<i>etext</i> — last locations in program	ConvexOS	END(3)
Euclidean distance	ConvexOS	HYPOT(3M)
evaluate arguments as an expression	ConvexOS	EXPR(1)
<i>ex</i> — Excelan 10 Mb/s Ethernet network interface	ConvexOS	EX(4)
<i>ex</i> — text editor	ConvexOS	EX(1)
examine and change signal action	ConvexOS	SIGACTION(2)
examine pending signals	ConvexOS	SIGPENDING(2)
Excelan 10 Mb/s Ethernet network interface	ConvexOS	EX(4)
<i>execl</i> — execute a file	ConvexOS	EXECL(3)
<i>execle</i> — execute a file	ConvexOS	EXECL(3)
<i>execlp</i> — execute a file	ConvexOS	EXECL(3)
<i>exec</i> — execute a file	ConvexOS	EXECL(3)
executable file format	ConvexOS	B.OUT(5)
execute a file	ConvexOS	EXECL(3)
execute a file	ConvexOS	EXECVE(2)
execute commands at a later time	ConvexOS	AT(1)
execute single SPU UNIX command from CONVEX UNIX	ConvexOS	SPUCMD(8)
execution accounting file	ConvexOS	ACCT(5)

Description	Product	Man Page
execution time profile	ConvexOS	PROFIL(2)
<i>execv</i> — execute a file	ConvexOS	EXECL(3)
<i>execve</i> — execute a file	ConvexOS	EXECVE(2)
<i>execvp</i> — execute a file	ConvexOS	EXECL(3)
<i>exit</i> — terminate a process after flushing any pending output	ConvexOS	EXIT(3)
<i>exp</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>expand</i> — expand tabs to spaces, and vice versa	ConvexOS	EXPAND(1)
expand tabs to spaces, and vice versa	ConvexOS	EXPAND(1)
<i>expf</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>explain</i> — print wordy sentences; thesaurus for diction	ConvexOS	DICTION(1)
<i>explain</i> — print wordy sentences; thesaurus for diction	ConvexOS	EXPLAIN(1)
exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>exports</i> — directories to export to NFS clients	ConvexOS	EXPORTS(5)
<i>expr</i> — evaluate arguments as an expression	ConvexOS	EXPR(1)
extract strings from C programs to implement shared strings	ConvexOS	XSTR(1)
<i>fabs</i> — absolute value, floor, ceiling functions	ConvexOS	FLOOR(3M)
<i>fabsf</i> — absolute value, floor, ceiling functions	ConvexOS	FLOOR(3M)
<i>faillog</i> — enable or disable logging of failed file accesses	ConvexOS	FAILLOG(2)
<i>faillogon</i> — log failed file access attempts	ConvexOS	FAILLOGON(8)
<i>faillogpr</i> — format the failed file access log	ConvexOS	FAILLOGPR(8)
<i>failure_log</i> — log of file access failures	ConvexOS	FAILURE_LOG(5)
<i>false</i> — provide truth values	ConvexOS	FALSE(1)
<i>false</i> — provide truth values	ConvexOS	TRUE(1)
fast incremental filesystem dump	ConvexOS	XDUMP(8)
<i>fastboot</i> — reboot/halt the system without checking the disks	ConvexOS	FASTBOOT(8)
<i>fasthalt</i> — reboot/halt the system without checking the disks	ConvexOS	FASTBOOT(8)
<i>fchmod</i> — change mode of file	ConvexOS	FCHMOD(2)
<i>fchown</i> — change owner and group of file by descriptor	ConvexOS	FCHOWN(2)
<i>fclose</i> — close or flush a stream	ConvexOS	FCLOSE(3S)
<i>fcntl</i> — file control	ConvexOS	FCNTL(2)
<i>fcvt</i> — output conversion	ConvexOS	ECVT(3)
<i>fdopen</i> — open a stream	ConvexOS	FOPEN(3S)
<i>fdpath</i> — return a path to a file from a file descriptor	ConvexOS	FDPATH(2)
<i>feof</i> — stream status inquiries	ConvexOS	FERROR(3S)
<i>ferror</i> — stream status inquiries	ConvexOS	FERROR(3S)
<i>fetch</i> — data base subroutines	ConvexOS	DBM(3X)
<i>fflush</i> — close or flush a stream	ConvexOS	FCLOSE(3S)
<i>ffs</i> — bit and byte string operations	ConvexOS	BSTRING(3)
<i>fgetc</i> — get character or word from stream	ConvexOS	GETC(3S)
<i>fgetgrent</i> — get group file entry	ConvexOS	GETGRENT(3)
<i>fgetpos</i> — reposition a stream	ConvexOS	FSEEK(3S)
<i>fgetpwent</i> — get password file entry	ConvexOS	GETPWENT(3)
<i>fgetpwrestent</i> — get entry from password restrictions file	ConvexOS	GETPWRESTENT(3)
<i>fgets</i> — get a string from a stream	ConvexOS	GETS(3S)

Description	Product	Man Page
<i>fgrep</i> — search a file for a pattern	ConvexOS	GREP(1)
<i>fhopen</i> — open a file for reading or writing via a file handle	ConvexOS	FHOPEN(2)
<i>fhpath</i> — return the path a file was opened with	ConvexOS	FHPATH(2)
<i>file</i> — determine file type	ConvexOS	FILE(1)
file control	ConvexOS	FCNTL(2)
file header for standard format object files	ConvexOS	FILEHDR(5)
file mode sticky bit	ConvexOS	STICKY(8)
file pager, alternative to more	ConvexOS	LESS(1)
file perusal filter for CRT viewing	ConvexOS	MORE(1)
file system consistency check and interactive repair	ConvexOS	FSCK(8)
file system directory consistency check	ConvexOS	DCHECK(8)
file system hierarchy	ConvexOS	HIER(7)
file system storage consistency check	ConvexOS	ICHECK(8)
<i>filehdr</i> — file header for standard format object files	ConvexOS	FILEHDR(5)
<i>fileno</i> — stream status inquiries	ConvexOS	FERROR(3S)
filter for Printronix printers	ConvexOS	PRX(1)
filter nroff output for CRT previewing	ConvexOS	COLCRT(1)
filter reverse line feeds	ConvexOS	COL(1)
<i>find</i> — find files	ConvexOS	FIND(1)
find files	ConvexOS	FIND(1)
find lines in a sorted list	ConvexOS	LOOK(1)
find name of a terminal	ConvexOS	TTYNAME(3)
find ordering relation for a 680x0 object library	ConvexOS	LORDER68(1)
find ordering relation for an object library	ConvexOS	LORDER(1)
find spelling errors	ConvexOS	SPELL(1)
find the printable strings in a object, or other binary, file	ConvexOS	STRINGS(1)
<i>finger</i> — user information lookup program	ConvexOS	FINGER(1)
<i>fingerd</i> — remote user information server	ConvexOS	FINGERD(8C)
<i>firstkey</i> — data base subroutines	ConvexOS	DBM(3X)
<i>float.h</i> — header file containing information on floating-point numbers	ConvexOS	FLOAT.H(3)
<i>flock</i> — apply or remove an advisory lock on an open file	ConvexOS	FLOCK(2)
<i>floor</i> — absolute value, floor, ceiling functions	ConvexOS	FLOOR(3M)
<i>fmod</i> — split into mantissa and exponent	ConvexOS	FREXP(3)
<i>fmt</i> — simple text formatter	ConvexOS	FMT(1)
<i>fold</i> — fold long lines for finite width output device	ConvexOS	FOLD(1)
fold long lines for finite width output device	ConvexOS	FOLD(1)
font formats for the Benson-Varian or Versatec	ConvexOS	VFONT(5)
<i>fopen</i> — open a stream	ConvexOS	FOPEN(3S)
<i>fork</i> — create a new process	ConvexOS	FORK(2)
format of an encoded uuencode file	ConvexOS	UUENCODE(5)
format of directories	ConvexOS	DIR(5)
format of file system volume	ConvexOS	FS(5)
format of memory image file	ConvexOS	CORE(5)
format of RCS file	ConvexOS	RCSFILE(5)
format tables for nroff	ConvexOS	TBL(1)

Description	Product	Man Page
format the failed file access log	ConvexOS	FAILLOGPR(8)
formatted input conversion	ConvexOS	SCANF(3S)
formatted output conversion	ConvexOS	PRINTF(3S)
<i>fpathconf</i> — configurable pathname variables	ConvexOS	PATHCONF(3)
<i>fprintf</i> — formatted output conversion	ConvexOS	PRINTF(3S)
<i>fputc</i> — put character or word on a stream	ConvexOS	PUTC(3S)
<i>fputs</i> — put a string on a stream	ConvexOS	PUTS(3S)
<i>fread</i> — buffered binary input/output	ConvexOS	FREAD(3S)
<i>free</i> — memory allocator	ConvexOS	MALLOC(3)
<i>freopen</i> — open a stream	ConvexOS	FOPEN(3S)
<i>frexp</i> — split into mantissa and exponent	ConvexOS	FREXP(3)
<i>from</i> — who is my mail from?	ConvexOS	FROM(1)
<i>fs</i> — format of file system volume	ConvexOS	FS(5)
<i>fscanf</i> — formatted input conversion	ConvexOS	SCANF(3S)
<i>fsck</i> — file system consistency check and interactive repair	ConvexOS	FSCK(8)
<i>fseek</i> — reposition a stream	ConvexOS	FSEEK(3S)
<i>fsetpos</i> — reposition a stream	ConvexOS	FSEEK(3S)
<i>fsrand</i> — install random inode generation numbers	ConvexOS	FSRAND(8)
<i>fstab</i> — static information about filesystems	ConvexOS	FSTAB(5)
<i>fstat</i> — get file status	ConvexOS	STAT(2)
<i>fstat</i> — identify open files	ConvexOS	FSTAT(8)
<i>fstatfs</i> — get file system statistics	ConvexOS	STATFS(2)
<i>fsync</i> — synchronize a file's in-memory state with that on disk	ConvexOS	FSYNC(2)
<i>ftell</i> — reposition a stream	ConvexOS	FSEEK(3S)
<i>ftime</i> — get formatted date and time	ConvexOS	FTIME(3C)
<i>ftpd</i> — DARPA Internet File Transfer Protocol server	ConvexOS	FTPD(8)
<i>ftruncate</i> — truncate a file to a specified length	ConvexOS	TRUNCATE(2)
functions for locale manipulation	ConvexOS	SETLOCALE(3)
<i>fwrite</i> — buffered binary input/output	ConvexOS	FREAD(3S)
<i>gamma</i> — log gamma function	ConvexOS	GAMMA(3M)
gateway data base for routed	ConvexOS	GATEWAYS(5)
<i>gateways</i> — gateway data base for routed	ConvexOS	GATEWAYS(5)
gather output to file	ConvexOS	WRITEV(2)
gather system statistics and print graphs of system statistics	ConvexOS	SEESTAT(8)
<i>gcvt</i> — output conversion	ConvexOS	ECVT(3)
general line printer filter	ConvexOS	LPF(8)
general terminal interface	ConvexOS	TERMIOS(4)
general terminal interface	ConvexOS	TTY(4)
generate a fault	ConvexOS	ABORT(3)
generate checksum for 680x0 b.out format file	ConvexOS	CS68(1)
generate encryption key	ConvexOS	MAKEKEY(8)
generate hashed password table(s)	ConvexOS	MKPASSWD(8)
generate names from i-numbers	ConvexOS	NCHECK(8)
generate password restrictions file	ConvexOS	GENREST(8)
generate terminal pathname	ConvexOS	CTERMID(3)
generator of lexical analysis programs	ConvexOS	LEX(1)

Description	Product	Man Page
<i>genrest</i> — generate password restrictions file	ConvexOS	GENREST(8)
get a process' activity ID	ConvexOS	GETAID(2)
get a process' real and effective group ID	ConvexOS	PGETREGID(2)
get a pseudo-teletype device name	ConvexOS	GETPTY(3)
get a string from a stream	ConvexOS	GETS(3S)
get activity file entry	ConvexOS	GETACTENT(3)
get actwho file entry	ConvexOS	GETACWENT(3)
get and set options on sockets	ConvexOS	GETSOCKOPT(2)
get character or word from stream	ConvexOS	GETC(3S)
get configurable system variables	ConvexOS	SYSCONF(3)
get current floating point mode	ConvexOS	GETFPMODE(3)
get current working directory pathname	ConvexOS	GETWD(3)
get descriptor table size	ConvexOS	GETDTABLESIZE(2)
get disk description by its name	ConvexOS	GETDISKBYNAME(3X)
get entries from name list	ConvexOS	NLIST(3)
get entry from password restrictions file	ConvexOS	GETPWRESTENT(3)
get file status	ConvexOS	CVXSTAT(2)
get file status	ConvexOS	LSTAT(2)
get file status	ConvexOS	STAT(2)
get file system descriptor file entry	ConvexOS	GETFSENT(3X)
get file system descriptor file entry	ConvexOS	GETMNTENT(3)
get file system statistics	ConvexOS	STATFS(2)
get formatted date and time	ConvexOS	FTIME(3C)
get group access list	ConvexOS	GETGROUPS(2)
get group file entry	ConvexOS	GETGRENT(3)
get group identity	ConvexOS	GETGID(2)
get information about parallel resource utilization	ConvexOS	CVXPRUSAGE(2)
get information about resource utilization	ConvexOS	GETRUSAGE(2)
get information about resource utilization	ConvexOS	VTIMES(3C)
get legal user shells	ConvexOS	GETUSERSHELL(3)
get login name	ConvexOS	GETLOGIN(3)
get name from uid	ConvexOS	GETPW(3C)
get name of connected peer	ConvexOS	GETPEERNAME(2)
get network entry	ConvexOS	GETNETENT(3N)
get network host entry	ConvexOS	GETHOSTBYNAME(3N)
get NIC format host tables	ConvexOS	GETTABLE(8C)
get option letter from argv	ConvexOS	GETOPT(3)
get password file entry	ConvexOS	GETPWENT(3)
get process group	ConvexOS	GETPGRP(2)
get process identification	ConvexOS	GETPID(2)
get process times	ConvexOS	TIMES(3C)
get processor time used	ConvexOS	CLOCK(3)
get protocol entry	ConvexOS	GETPROTOENT(3N)
get service entry	ConvexOS	GETSERVENT(3N)
get shares file entry given uid	ConvexOS	GETSHARES(3)
get socket name	ConvexOS	GETSOCKNAME(2)
get system information	ConvexOS	UNAME(2)
get system information	ConvexOS	GETSYSINFO(2)
get system page size	ConvexOS	GETPAGESIZE(2)

<b>Description</b>	<b>Product</b>	<b>Man Page</b>
get system time	ConvexOS	TIME(3C)
get terminal name	ConvexOS	TTY(1)
get the current working directory for the process	ConvexOS	GETCWD(3)
get ttys file entry	ConvexOS	GETTTYENT(3)
get user identity	ConvexOS	GETUID(2)
get user name	ConvexOS	CUSERID(3)
<i>getactaid</i> — get activity file entry	ConvexOS	GETACTENT(3)
<i>getactent</i> — get activity file entry	ConvexOS	GETACTENT(3)
<i>getactnam</i> — get activity file entry	ConvexOS	GETACTENT(3)
<i>getacwent</i> — get actwho file entry	ConvexOS	GETACWENT(3)
<i>getaid</i> — get a process' activity ID	ConvexOS	GETAID(2)
<i>getc</i> — get character or word from stream	ConvexOS	GETC(3S)
<i>getchar</i> — get character or word from stream	ConvexOS	GETC(3S)
<i>getcwd</i> — get the current working directory for the process	ConvexOS	GETCWD(3)
<i>getdirentries</i> — gets directory entries in a filesystem independent format	ConvexOS	GETDIRENTRIES(2)
<i>getdiskbyname</i> — get disk description by its name	ConvexOS	GETDISKBYNAME(3X)
<i>getdomainname</i> — get/set name of current domain	ConvexOS	GETDOMAINNAME(2)
<i>getdtablesize</i> — get descriptor table size	ConvexOS	GETDTABLESIZE(2)
<i>getegid</i> — get group identity	ConvexOS	GETGID(2)
<i>getenv</i> — manipulate environmental variables	ConvexOS	GETENV(3)
<i>geteuid</i> — get user identity	ConvexOS	GETUID(2)
<i>getfpmode</i> — get current floating point mode	ConvexOS	GETFPMODE(3)
<i>getfsent</i> — get file system descriptor file entry	ConvexOS	GETFSENT(3X)
<i>getfsfile</i> — get file system descriptor file entry	ConvexOS	GETFSENT(3X)
<i>getfsspec</i> — get file system descriptor file entry	ConvexOS	GETFSENT(3X)
<i>getfstype</i> — get file system descriptor file entry	ConvexOS	GETFSENT(3X)
<i>getgid</i> — get group identity	ConvexOS	GETGID(2)
<i>getgrent</i> — get group file entry	ConvexOS	GETGRENT(3)
<i>getgrgid</i> — group database access routines	ConvexOS	GETGRGID(3)
<i>getgrnam</i> — group database access routines	ConvexOS	GETGRGID(3)
<i>getgroups</i> — get group access list	ConvexOS	GETGROUPS(2)
<i>gethostbyaddr</i> — get network host entry	ConvexOS	GETHOSTBYNAME(3N)
<i>gethostbyname</i> — get network host entry	ConvexOS	GETHOSTBYNAME(3N)
<i>gethostent</i> — get network host entry	ConvexOS	GETHOSTBYNAME(3N)
<i>gethostid</i> — get/set unique identifier of current host	ConvexOS	GETHOSTID(2)
<i>gethostname</i> — get/set name of current host	ConvexOS	GETHOSTNAME(2)
<i>getitimer</i> — get/set value of interval timer	ConvexOS	GETITIMER(2)
<i>getlogin</i> — get login name	ConvexOS	GETLOGIN(3)
<i>getlogin</i> — get user name	ConvexOS	CUSERID(3)
<i>getmntent</i> — get file system descriptor file entry	ConvexOS	GETMNTENT(3)
<i>getnetbyaddr</i> — get network entry	ConvexOS	GETNETENT(3N)
<i>getnetbyname</i> — get network entry	ConvexOS	GETNETENT(3N)
<i>getnetent</i> — get network entry	ConvexOS	GETNETENT(3N)
<i>getopt</i> — get option letter from argv	ConvexOS	GETOPT(3)
<i>getpagesize</i> — get system page size	ConvexOS	GETPAGESIZE(2)
<i>getpass</i> — read a password	ConvexOS	GETPASS(3)
<i>getpattr</i> — get/set process attributes	ConvexOS	GETPATR(2)

Description	Product	Man Page
<i>getpeername</i> — get name of connected peer	ConvexOS	GETPEERNAME(2)
<i>getpflags</i> — get/set process entry flags	ConvexOS	GETPFLAGS(2)
<i>getpgrp</i> — get process group	ConvexOS	GETPGRP(2)
<i>getpid</i> — get process identification	ConvexOS	GETPID(2)
<i>getppid</i> — get process identification	ConvexOS	GETPID(2)
<i>getpriority</i> — get/set program scheduling priority	ConvexOS	GETPRIORITY(2)
<i>getprotobyname</i> — get protocol entry	ConvexOS	GETPROTOENT(3N)
<i>getprotobynumber</i> — get protocol entry	ConvexOS	GETPROTOENT(3N)
<i>getprotoent</i> — get protocol entry	ConvexOS	GETPROTOENT(3N)
<i>getpty</i> — get a pseudo-teletype device name	ConvexOS	GETPTY(3)
<i>getpw</i> — get name from uid	ConvexOS	GETPW(3C)
<i>getpwent</i> — get password file entry	ConvexOS	GETPWENT(3)
<i>getpwnam</i> — user database access	ConvexOS	GETPWUID(3)
<i>getpwrestent</i> — get entry from password restrictions file	ConvexOS	GETPWRESTENT(3)
<i>getpwrestnam</i> — get entry from password restrictions file	ConvexOS	GETPWRESTENT(3)
<i>getpwrestuid</i> — get entry from password restrictions file	ConvexOS	GETPWRESTENT(3)
<i>getpwuid</i> — user database access	ConvexOS	GETPWUID(3)
<i>getrlimit</i> — control maximum system resource consumption	ConvexOS	GETRLIMIT(2)
<i>getrusage</i> — get information about resource utilization	ConvexOS	GETRUSAGE(2)
<i>gets</i> — get a string from a stream	ConvexOS	GETS(3S)
<i>gets</i> directory entries in a filesystem independent format	ConvexOS	GETDIRENTRIES(2)
<i>getservbyname</i> — get service entry	ConvexOS	GETSERVENT(3N)
<i>getservbyport</i> — get service entry	ConvexOS	GETSERVENT(3N)
<i>getservent</i> — get service entry	ConvexOS	GETSERVENT(3N)
<i>getshares</i> — get shares file entry given uid	ConvexOS	GETSHARES(3)
<i>getshput</i> — read, modify, and write shares file entry	ConvexOS	GETSHPUT(3)
<i>getsockname</i> — get socket name	ConvexOS	GETSOCKNAME(2)
<i>getsockopt</i> — get and set options on sockets	ConvexOS	GETSOCKOPT(2)
<i>getst</i> — print information from the kernel's stripe table	ConvexOS	GETST(8)
<i>getsysinfo</i> — get system information	ConvexOS	GETSYSINFO(2)
<i>getsysinfo</i> — print out system information	ConvexOS	GETSYSINFO(1)
<i>gettable</i> — get NIC format host tables	ConvexOS	GETTABLE(8C)
<i>gettimeofday</i> — get/set date and time	ConvexOS	GETTIMEOFDAY(2)
<i>getttyent</i> — get ttys file entry	ConvexOS	GETTTYENT(3)
<i>getttynam</i> — get ttys file entry	ConvexOS	GETTTYENT(3)
<i>getty</i> — set terminal mode	ConvexOS	GETTY(8)
<i>gettytab</i> — terminal configuration data base	ConvexOS	GETTYTAB(5)
<i>getuid</i> — get user identity	ConvexOS	GETUID(2)
<i>getusershell</i> — get legal user shells	ConvexOS	GETUSERSHELL(3)
<i>getw</i> — get character or word from stream	ConvexOS	GETC(3S)
<i>getwd</i> — get current working directory pathname	ConvexOS	GETWD(3)
give first few lines	ConvexOS	HEAD(1)
<i>gmtime</i> — date and time manipulation routines	ConvexOS	CTIME(3)
GNU project Emacs	ConvexOS	EMACS(1)

<b>Description</b>	<b>Product</b>	<b>Man Page</b>
graphics interface	ConvexOS	PLOT(3X)
graphics interface	ConvexOS	PLOT(5)
<i>grep</i> — search a file for a pattern	ConvexOS	GREP(1)
<i>group</i> — group file	ConvexOS	GROUP(5)
group database access routines	ConvexOS	GETGRGID(3)
group file	ConvexOS	GROUP(5)
group-activity access control file	ConvexOS	ACTWHO(5)
<i>groups</i> — show group memberships	ConvexOS	GROUPS(1)
<i>gtty</i> — set and get terminal state (defunct)	ConvexOS	STTY(3C)
<i>gut</i> — remove text/data/bss sections from an executable	ConvexOS	GUT(1)
<i>halt</i> — stop the processor	ConvexOS	HALT(8)
reboot/halt the system without checking the disks	ConvexOS	FASTBOOT(8)
handle remote mail received via uucp	ConvexOS	RMMAIL(1)
<i>hasmntopt</i> — get file system descriptor file entry	ConvexOS	GETMNTENT(3)
HASP Host communications boards database	ConvexOS	CURRENT_RMETS(5)
HASP Host communications capability database	ConvexOS	COMMCAP(5)
<i>hconfig</i> — Maintains the HASP Host communications capability database	ConvexOS	HCONFIG(8)
<i>head</i> — give first few lines	ConvexOS	HEAD(1)
header file contain numerical limits	ConvexOS	LIMITS.H(3)
header file containing information on floating-point numbers	ConvexOS	FLOAT.H(3)
header file contains standard definitions	ConvexOS	STDDEF.H(3)
header file relating to error reporting	ConvexOS	ERRNO.H(3)
<i>help</i> — online information system	ConvexOS	HELP(1)
<i>herror</i> — get network host entry	ConvexOS	GETHOSTBYNAME(3N)
<i>hhstart</i> — Start HASP Host	ConvexOS	HHSTART(8)
<i>hhstop</i> — Stop HASP Host	ConvexOS	HHSTOP(8)
<i>hier</i> — file system hierarchy	ConvexOS	HIER(7)
host name data base	ConvexOS	HOSTS(5)
host name resolution description	ConvexOS	HOSTNAME(7)
<i>hostid</i> — set or print identifier of current host system	ConvexOS	HOSTID(1)
<i>hostname</i> — host name resolution description	ConvexOS	HOSTNAME(7)
<i>hostname</i> — set or print name of current host system	ConvexOS	HOSTNAME(1)
<i>hosts</i> — host name data base	ConvexOS	HOSTS(5)
<i>hosts.equiv</i> — list of trusted hosts, remote host access file	ConvexOS	HOSTS.EQUIV(5)
<i>hsend</i> — Transfer file from a Convex computer to a remote workstation	ConvexOS	HSEND(8)
<i>htable</i> — convert NIC standard format host tables	ConvexOS	HTABLE(8C)
<i>htonl</i> — convert values between host and network byte order	ConvexOS	BYTEORDER(3N)
<i>htons</i> — convert values between host and network byte order	ConvexOS	BYTEORDER(3N)
<i>hy</i> — HYPERchannel driver or network interface	ConvexOS	HY(4)
hyperbolic functions	ConvexOS	SINH(3M)
HYPERchannel driver or network interface	ConvexOS	HY(4)
<i>hypot</i> — Euclidean distance	ConvexOS	HYPOT(3M)

Description	Product	Man Page
<i>hyroute</i> — set the HYPERchannel routing tables	ConvexOS	HYROUTE(8C)
<i>hystat</i> — acquire and display HYPERchannel adapter statistics	ConvexOS	HYSTAT(8C)
<i>icheck</i> — file system storage consistency check	ConvexOS	ICHECK(8)
<i>icmp</i> — Internet Control Message Protocol	ConvexOS	ICMP(4P)
<i>idcvtd</i> — IEEE/native floating point mode conversion routines.	ConvexOS	RCVTIR(3M)
<i>ident</i> — identify files	ConvexOS	IDENT(1)
identify files	ConvexOS	IDENT(1)
identify open files	ConvexOS	FSTAT(8)
<i>idle</i> — display idle time	ConvexOS	IDLE(1)
<i>idtoname</i> — numeric ID to name filter	ConvexOS	IDTONAME(1)
<i>ifconfig</i> — configure network interface parameters	ConvexOS	IFCONFIG(8c)
lkon controller for Versatec plotters	ConvexOS	PB(4)
<i>in.comsat</i> — biff server	ConvexOS	COMSAT(8C)
<i>in.rexecd</i> — remote execution server	ConvexOS	REXECD(8C)
<i>in.rlogind</i> — remote login server	ConvexOS	RLOGIND(8C)
<i>in.rshd</i> — remote shell server	ConvexOS	RSHD(8C)
<i>in.talkd</i> — talk daemon	ConvexOS	TALKD(8)
incremental dump format	ConvexOS	DUMP(5)
incremental file system restore	ConvexOS	RESTORE(8)
incremental filesystem dump	ConvexOS	DUMP(8)
<i>indent</i> — indent and format C program source	ConvexOS	INDENT(1)
indent and format C program source	ConvexOS	INDENT(1)
<i>index</i> — string operations	ConvexOS	STRING(3)
<i>index</i> — string search functions	ConvexOS	STRINGSEARCH(3)
indicate last logins of users and teletypes	ConvexOS	LAST(1)
indirect system call	ConvexOS	SYSCALL(2)
indivisibly test and set a memory location	ConvexOS	TAS(3)
<i>inet</i> — Internet protocol family	ConvexOS	INET(4F)
<i>inet_addr</i> — Internet address manipulation routines	ConvexOS	INET(3N)
<i>inet_lnaof</i> — Internet address manipulation routines	ConvexOS	INET(3N)
<i>inet_makeaddr</i> — Internet address manipulation routines	ConvexOS	INET(3N)
<i>inet_netof</i> — Internet address manipulation routines	ConvexOS	INET(3N)
<i>inet_network</i> — Internet address manipulation routines	ConvexOS	INET(3N)
<i>inet_ntoa</i> — Internet address manipulation routines	ConvexOS	INET(3N)
<i>inetd</i> — internet super-server daemon and configuration file	ConvexOS	INETD(8C)
<i>inetd.conf</i> — internet super-server daemon and configuration file	ConvexOS	INETD(8C)
<i>info</i> — menu driven online information system	ConvexOS	INFO(1)
<i>init</i> — process control initialization	ConvexOS	INIT(8)
<i>initgroups</i> — initialize group access list	ConvexOS	INITGROUPS(3X)
initialize group access list	ConvexOS	INITGROUPS(3X)
initiate a connection on a socket	ConvexOS	CONNECT(2)
initiate a UUCP phone call to a neighboring site	ConvexOS	SEND(1)
initiate I/O to/from a process or pipe	ConvexOS	POPEN(3)
<i>initstate</i> — better random number generator; routines for changing generators	ConvexOS	RANDOM(3)

Description	Product	Man Page
<i>inline</i> — inline subroutine call expander for kernel builds	ConvexOS	INLINE(8)
inline subroutine call expander for kernel builds	ConvexOS	INLINE(8)
<i>inode</i> — format of file system volume	ConvexOS	FS(5)
insert articles into a notesfile	ConvexOS	NFPIPE(1)
<i>insque</i> — insert/remove element from a queue	ConvexOS	INSQUE(3)
<i>install</i> — install binaries	ConvexOS	INSTALL(1)
install binaries	ConvexOS	INSTALL(1)
install or create CONVEX software release tapes	ConvexOS	INSTALLSW(8)
install random inode generation numbers	ConvexOS	FSRAND(8)
<i>installsw</i> — install or create CONVEX software release tapes	ConvexOS	INSTALLSW(8)
Integrated Disk Controller IPI-2 Logical Level for Disk special file	ConvexOS	DU(4)
Internet address manipulation routines	ConvexOS	INET(3N)
Internet Control Message Protocol	ConvexOS	ICMP(4P)
Internet domain name server	ConvexOS	NAMED(8)
Internet Protocol	ConvexOS	IP(4P)
Internet protocol family	ConvexOS	INET(4F)
internet super-server daemon and configuration file	ConvexOS	INETD(8C)
Internet Transmission Control Protocol	ConvexOS	TCP(4P)
Internet User Datagram Protocol	ConvexOS	UDP(4P)
<i>intro</i> — introduction to commands	ConvexOS	INTRO(1)
<i>intro</i> — introduction to compatibility library functions	ConvexOS	INTRO(3C)
<i>intro</i> — introduction to library functions	ConvexOS	INTRO(3)
<i>intro</i> — introduction to mathematical library functions	ConvexOS	INTRO(3M)
<i>intro</i> — introduction to miscellaneous library functions	ConvexOS	INTRO(3X)
<i>intro</i> — introduction to network library functions	ConvexOS	INTRO(3N)
<i>intro</i> — introduction to special files and hardware support	ConvexOS	INTRO(4)
<i>intro</i> — introduction to system calls and error numbers	ConvexOS	INTRO(2)
<i>intro</i> — introduction to system maintenance and operation commands	ConvexOS	INTRO(8)
introduction to commands	ConvexOS	INTRO(1)
introduction to compatibility library functions	ConvexOS	INTRO(3C)
introduction to library functions	ConvexOS	INTRO(3)
introduction to mathematical library functions	ConvexOS	INTRO(3M)
introduction to miscellaneous library functions	ConvexOS	INTRO(3X)
introduction to network library functions	ConvexOS	INTRO(3N)
introduction to networking facilities	ConvexOS	INTRO(4N)
introduction to special files and hardware support	ConvexOS	INTRO(4)
introduction to system calls and error numbers	ConvexOS	INTRO(2)
introduction to system maintenance and operation commands	ConvexOS	INTRO(8)
<i>ioconfig</i> — System I/O configuration description file	ConvexOS	IOCONFIG(4)
<i>ioctl</i> — control device	ConvexOS	IOCTL(2)
<i>iostats</i> — I/O statistics	ConvexOS	IOSTATS(4)
<i>ip</i> — Internet Protocol	ConvexOS	IP(4P)
<i>ipow</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)

Description	Product	Man Page
<i>ircvtr</i> — IEEE/native floating point mode conversion routines.	ConvexOS	RCVTIR(3M)
<i>isalnum</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>isalpha</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>isascii</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>isatty</i> — find name of a terminal	ConvexOS	TTYNAME(3)
<i>isctrl</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>isdigit</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>isgraph</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>islower</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>isprint</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>ispunct</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>isspace</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
issue a shell command	ConvexOS	SYSTEM(3)
<i>isupper</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>iszdigit</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>j0</i> — bessel functions	ConvexOS	J0(3M)
<i>j1</i> — bessel functions	ConvexOS	J0(3M)
<i>jn</i> — bessel functions	ConvexOS	J0(3M)
<i>join</i> — relational database operator	ConvexOS	JOIN(1)
kernel user shares structure	ConvexOS	LNODE(5)
<i>kill</i> — send signal to a process	ConvexOS	KILL(2)
<i>kill</i> — terminate a process with extreme prejudice	ConvexOS	KILL(1)
<i>killpg</i> — send signal to a process group	ConvexOS	KILLPG(2)
<i>kmem</i> — main memory	ConvexOS	MEM(4)
<i>krpc</i> — daemon interface to kernel RPC facility	ConvexOS	KRPC(2)
<i>L.aliases</i> — UUCP hostname alias file	ConvexOS	L.ALIASES(5)
<i>L.cmds</i> — UUCP remote command permissions file	ConvexOS	L.CMDS(5)
<i>L.sys</i> — UUCP remote host description file	ConvexOS	L.SYS(5)
<i>label</i> — graphics interface	ConvexOS	PLOT(3X)
<i>labs</i> — standard integral functions	ConvexOS	ABS(3)
<i>last</i> — indicate last logins of users and teletypes	ConvexOS	LAST(1)
last locations in program	ConvexOS	END(3)
<i>lastcomm</i> — show last commands executed in reverse order	ConvexOS	LASTCOMM(1)
<i>ld</i> — link editor	ConvexOS	LD(1)
<i>ld68</i> — 680x0 link editor	ConvexOS	LD68(1)
<i>ld88</i> — MC88100 Link Editor	ConvexOS	LD88(1)
<i>ldexp</i> — split into mantissa and exponent	ConvexOS	FREXP(3)
<i>ldiv</i> — standard integral functions	ConvexOS	ABS(3)
<i>learn</i> — computer aided instruction about UNIX	ConvexOS	LEARN(1)
<i>leave</i> — remind you when you have to leave	ConvexOS	LEAVE(1)
<i>less</i> — file pager, alternative to more	ConvexOS	LESS(1)
<i>lesskey</i> — specify key bindings for less	ConvexOS	LESSKEY(1)
<i>lex</i> — generator of lexical analysis programs	ConvexOS	LEX(1)
<i>lf</i> — list contents of directory	ConvexOS	LS(1)
<i>lim</i> — change share data base information for one or more users	ConvexOS	LIM(8)
<i>limits</i> — return or set limits or cost structures	ConvexOS	LIMITS(2)

Description	Product	Man Page
<i>limits.h</i> — header file contain numerical limits	ConvexOS	LIMITS.H(3)
<i>line</i> — graphics interface	ConvexOS	PLOT(3X)
line printer control program	ConvexOS	LPC(8)
line printer daemon	ConvexOS	LPD(8)
line printer driver	ConvexOS	PA(4)
line printer rewind commands	ConvexOS	LPREWIND(8)
<i>linemod</i> — graphics interface	ConvexOS	PLOT(3X)
<i>link</i> — make a hard link to a file	ConvexOS	LINK(2)
link editor	ConvexOS	LD(1)
<i>lint</i> — C program verifier	ConvexOS	LINT(1)
<i>lint68</i> — C program verifier	ConvexOS	LINT68(1)
list contents of directory	ConvexOS	LS(1)
list label information on a labeled tape	ConvexOS	TPLIST(1)
list of operator mnemonics and restrictions enforced by the op program	ConvexOS	OP.ACCESS(5)
list of trusted hosts, remote host access file	ConvexOS	HOSTS.EQUIV(5)
<i>listen</i> — listen for connections on a socket	ConvexOS	LISTEN(2)
listen for connections on a socket	ConvexOS	LISTEN(2)
<i>ll</i> — list contents of directory	ConvexOS	LS(1)
<i>ln</i> — make links	ConvexOS	LN(1)
<i>lnode</i> — kernel user shares structure	ConvexOS	LNODE(5)
<i>lo</i> — software loopback network interface	ConvexOS	LO(4)
<i>localeconv</i> — functions for locale manipulation	ConvexOS	SETLOCALE(3)
<i>localtime</i> — date and time manipulation routines	ConvexOS	CTIME(3)
locate a program file including aliases and paths (csh only)	ConvexOS	WHICH(1)
locate commands by keyword lookup	ConvexOS	APROPOS(1)
<i>lock</i> — reserve a terminal	ConvexOS	LOCK(1)
<i>lockd</i> — network lock daemon	ConvexOS	LOCKD(8C)
<i>lockf</i> — advisory record locking on files	ConvexOS	LOCKF(3)
<i>log</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
log failed file access attempts	ConvexOS	FAILLOGON(8)
log gamma function	ConvexOS	GAMMA(3M)
log generated by bill command	ConvexOS	BILL-ACCT(5)
log of failed login attempts	ConvexOS	BADLOGINS(5)
log of file access failures	ConvexOS	FAILURE_LOG(5)
log systems messages	ConvexOS	SYSLOGD(8)
<i>log10</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>log10f</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>logf</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>logger</i> — make entries in the system log	ConvexOS	LOGGER(1)
<i>login</i> — sign on	ConvexOS	LOGIN(1)
login accounting	ConvexOS	AC(8)
login records	ConvexOS	UTMP(5)
<i>longjmp</i> — non-local goto	ConvexOS	SETJMP(3)
<i>look</i> — find lines in a sorted list	ConvexOS	LOOK(1)
<i>lorder</i> — find ordering relation for an object library	ConvexOS	LORDER(1)
<i>lorder68</i> — find ordering relation for a 680x0 object library	ConvexOS	LORDER68(1)

Description	Product	Man Page
<i>lpc</i> — line printer control program	ConvexOS	LPC(8)
<i>lpd</i> — line printer daemon	ConvexOS	LPD(8)
<i>lpf</i> — general line printer filter	ConvexOS	LPF(8)
<i>lpmv</i> — move jobs from one line printer spooling queue to another queue	ConvexOS	LPMV(1)
<i>lpow</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>lpq</i> — spool queue examination program	ConvexOS	LPQ(1)
<i>lpr</i> — off line print	ConvexOS	LPR(1)
<i>lprm</i> — remove jobs from the line printer spooling queue	ConvexOS	LPRM(1)
<i>lprman</i> — process nroff-style tty37 output for Printronix printers	ConvexOS	LPRMAN(1)
<i>lprofil</i> — execution time profile	ConvexOS	PROFIL(2)
<i>lprx</i> — process nroff-style tty37 output for Printronix printers	ConvexOS	LPRMAN(1)
<i>lr</i> — list contents of directory	ConvexOS	LS(1)
<i>ls</i> — list contents of directory	ConvexOS	LS(1)
<i>lseek</i> — move read/write pointer	ConvexOS	LSEEK(2)
<i>lstat</i> — get file status	ConvexOS	LSTAT(2)
<i>m4</i> — macro processor	ConvexOS	M4(1)
macro processor	ConvexOS	M4(1)
macros for formatting papers	ConvexOS	ME(7)
macros to typeset manual	ConvexOS	MAN(7)
<i>magic</i> — data base of magic number checks used by the file(1) utility	ConvexOS	MAGIC(5)
magnetic tape manipulating program	ConvexOS	MT(1)
<i>mail</i> — send and receive mail	ConvexOS	MAIL(1)
mail addressing description	ConvexOS	MAILADDR(7)
<i>mailaddr</i> — mail addressing description	ConvexOS	MAILADDR(7)
main memory	ConvexOS	MEM(4)
maintain and process system availability information	ConvexOS	AVAIL(8)
maintain program groups	ConvexOS	MAKE(1)
Maintains the HASP Host communications capability database	ConvexOS	HCONFIG(8)
<i>make</i> — maintain program groups	ConvexOS	MAKE(1)
make a directory	ConvexOS	MKDIR(1)
make a directory file	ConvexOS	MKDIR(2)
make a FIFO special file	ConvexOS	MKFIFO(3)
make a hard link to a file	ConvexOS	LINK(2)
make a lost+found directory for fsck	ConvexOS	MKLOST+FOUND(8)
make a new striped filesystem	ConvexOS	NEWST(8)
make a special (character, block, or fifo) file	ConvexOS	MKNOD(2)
make a unique file name	ConvexOS	MKTEMP(3)
make entries in the system log	ConvexOS	LOGGER(1)
make links	ConvexOS	LN(1)
make symbolic link to a file	ConvexOS	SYMLINK(2)
make system special files	ConvexOS	MAKEDEV(8)
make typescript of terminal session	ConvexOS	SCRIPT(1)
<i>makedev</i> — make system special files	ConvexOS	MAKEDEV(8)

Description	Product	Man Page
<i>MAKEDEV</i> — make system special files	ConvexOS	MAKEDEV(8)
<i>makekey</i> — generate encryption key	ConvexOS	MAKEKEY(8)
<i>makewhatis</i> — rebuild the whatis database	ConvexOS	MAKEWHATIS(8)
<i>malloc</i> — memory allocator	ConvexOS	MALLOC(3)
<i>man</i> — display on-line reference manual information	ConvexOS	MAN(1)
<i>man</i> — macros to typeset manual	ConvexOS	MAN(7)
manipulate disk quotas	ConvexOS	QUOTACTL(2)
manipulate environmental variables	ConvexOS	GETENV(3)
manipulate kernel databases for striped disk partitions	ConvexOS	PUTST(8)
manipulate signal sets	ConvexOS	SIGSETOPS(3)
manually manipulate the routing tables	ConvexOS	ROUTE(8C)
map of ASCII character set	ConvexOS	ASCII(7)
map shared memory into your address space	ConvexOS	MMAP(2)
mass storage disk interface	ConvexOS	DA(4)
<i>mblen</i> — multibyte and wide character functions	ConvexOS	MBSTOWCS(3)
<i>mbstowcs</i> — multibyte and wide character functions	ConvexOS	MBSTOWCS(3)
<i>mbtowc</i> — multibyte and wide character functions	ConvexOS	MBSTOWCS(3)
MC88100 C Preprocessor	ConvexOS	CPP88(1)
MC88100 Link Editor	ConvexOS	LD88(1)
MC88100 print name list	ConvexOS	NM88(1)
<i>mclear</i> — shared memory synchronization primitives	ConvexOS	MSET(3)
<i>me</i> — macros for formatting papers	ConvexOS	ME(7)
<i>mem</i> — main memory	ConvexOS	MEM(4)
<i>memchr</i> — string search functions	ConvexOS	STRINGSEARCH(3)
<i>memcmp</i> — string comparison functions	ConvexOS	STRINGCMP(3)
<i>memcpy</i> — string copy functions	ConvexOS	STRINGCPY(3)
<i>memmove</i> — string copy functions	ConvexOS	STRINGCPY(3)
memory allocator	ConvexOS	MALLOC(3)
<i>memset</i> — string copy functions	ConvexOS	STRINGCPY(3)
menu driven online information system	ConvexOS	INFO(1)
<i>merge</i> — three-way file merge	ConvexOS	MERGE(1)
merge RCS revisions	ConvexOS	RCSMERGE(1)
<i>mesg</i> — permit or deny messages	ConvexOS	MSG(1)
<i>miscellaneous</i> — miscellaneous useful information pages	ConvexOS	INTRO(7)
miscellaneous useful information pages	ConvexOS	INTRO(7)
<i>mkdep</i> — construct Makefile dependency list	ConvexOS	MKDEP(1)
<i>mkdir</i> — make a directory	ConvexOS	MKDIR(1)
<i>mkdir</i> — make a directory file	ConvexOS	MKDIR(2)
<i>mkfifo</i> — make a FIFO special file	ConvexOS	MKFIFO(3)
<i>mkfs</i> — construct a file system	ConvexOS	MKFS(8)
<i>mklost+found</i> — make a lost+found directory for fsck	ConvexOS	MKLOST+FOUND(8)
<i>mknf</i> — create and delete notesfiles	ConvexOS	MKNF(8)
<i>mknod</i> — build special file	ConvexOS	MKNOD(8)
<i>mknod</i> — make a special (character, block, or fifo) file	ConvexOS	MKNOD(2)
<i>mkpasswd</i> — generate hashed password table(s)	ConvexOS	MKPASSWD(8)
<i>mkstemp</i> — make a unique file name	ConvexOS	MKTEMP(3)
<i>mkstr</i> — create an error message file by massaging C source	ConvexOS	MKSTR(1)

Description	Product	Man Page
<i>mktemp</i> — make a unique file name	ConvexOS	MKTEMP(3)
<i>mktime</i> — date and time manipulation routines	ConvexOS	CTIME(3)
<i>mmap</i> — map shared memory into your address space	ConvexOS	MMAP(2)
<i>mntent</i> — static information about filesystems	ConvexOS	MNTENT(5)
<i>modf</i> — split into mantissa and exponent	ConvexOS	FREXP(3)
modify process attributes	ConvexOS	MPA(1)
<i>moncontrol</i> — prepare execution profile	ConvexOS	MONITOR(3)
<i>monitor</i> — prepare execution profile	ConvexOS	MONITOR(3)
monitor inter-system communications	ConvexOS	UULOOK(8)
<i>monstartup</i> — prepare execution profile	ConvexOS	MONITOR(3)
<i>more</i> — file perusal filter for CRT viewing	ConvexOS	MORE(1)
<i>mount</i> — mount and dismount filesystems	ConvexOS	MOUNT(8)
<i>mount</i> — mount file system	ConvexOS	MOUNT(2)
mount and dismount filesystems	ConvexOS	MOUNT(8)
mount file system	ConvexOS	MOUNT(2)
mounted file system table	ConvexOS	MTAB(5)
<i>move</i> — graphics interface	ConvexOS	PLOT(3X)
move jobs from one line printer spooling queue to another queue	ConvexOS	LPMV(1)
move or rename files	ConvexOS	MV(1)
move read/write pointer	ConvexOS	LSEEK(2)
<i>mpa</i> — modify process attributes	ConvexOS	MPA(1)
<i>mremap</i> — change attributes of a memory region in a process' address space	ConvexOS	MREMAP(2)
<i>ms</i> — text formatting macros	ConvexOS	MS(7)
<i>mset</i> — shared memory synchronization primitives	ConvexOS	MSET(3)
<i>msgs</i> — system messages and junk mail program	ConvexOS	MSGS(1)
<i>msleep</i> — shared memory synchronization primitives	ConvexOS	MSLEEP(2)
<i>msync</i> — synchronize shared memory segment with file system	ConvexOS	MSYNC(2)
<i>mt</i> — magnetic tape manipulating program	ConvexOS	MT(1)
<i>/etc/mtab</i> — mounted file system table	ConvexOS	MTAB(5)
<i>mtio</i> — UNIX magtape interface	ConvexOS	MTIO(4)
multibyte and wide character functions	ConvexOS	MBSTOWCS(3)
<i>munmap</i> — unmap memory	ConvexOS	MUNMAP(2)
<i>mv</i> — move or rename files	ConvexOS	MV(1)
<i>mwakeup</i> — shared memory synchronization primitives	ConvexOS	MSLEEP(2)
<i>named</i> — Internet domain name server	ConvexOS	NAMED(8)
IEEE/native floating point mode conversion routines.	ConvexOS	RCVTIR(3M)
<i>ncheck</i> — generate names from i-numbers	ConvexOS	NCHECK(8)
<i>neqn</i> — typeset mathematics	ConvexOS	NEQN(1)
network lock daemon	ConvexOS	LOCKD(8C)
network name data base	ConvexOS	NETWORKS(5)
network routing daemon	ConvexOS	ROUTED(8C)
network status monitor	ConvexOS	STATD(8C)
<i>networking</i> — introduction to networking facilities	ConvexOS	INTRO(4N)
<i>networks</i> — network name data base	ConvexOS	NETWORKS(5)
<i>newaliases</i> — rebuild the data base for the mail aliases file	ConvexOS	NEWALIASES(1)

Description	Product	Man Page
<i>newfs</i> — construct a new file system	ConvexOS	NEWFS(8)
news system	ConvexOS	NOTES(1)
<i>newst</i> — make a new striped filesystem	ConvexOS	NEWST(8)
<i>nextkey</i> — data base subroutines	ConvexOS	DBM(3X)
<i>nfabort</i> — dump core and log it in a notesfile	ConvexOS	NFABORT(3)
<i>nfaccess</i> — add access rights to a set of Notesfiles	ConvexOS	NFACCESS(8)
<i>nfarchive</i> — archive notesfiles	ConvexOS	NFARCHIVE(8)
<i>nfcomment</i> — user interface to the notesfile system	ConvexOS	NFCOMMENT(3)
<i>nfcoun</i> — print number of notes with/without director's messages in a notesfile	ConvexOS	NFCOUNT(1)
<i>nfdump</i> — notesfile dump/load programs	ConvexOS	NFDUMP(8)
<i>nfload</i> — notesfile dump/load programs	ConvexOS	NFDUMP(8)
<i>nfmail</i> — accept mail for a notesfile	ConvexOS	NFMAIL(8)
<i>nfpipe</i> — insert articles into a notesfile	ConvexOS	NFPIPE(1)
<i>nfprint</i> — print the contents of a notesfile	ConvexOS	NFPRINT(1)
<i>nfrcv</i> — notesfile networking programs	ConvexOS	NFXMIT(8)
<i>nfstats</i> — print statistics about Notesfiles	ConvexOS	NFSTATS(1)
<i>nfzmit</i> — notesfile networking programs	ConvexOS	NFXMIT(8)
<i>nice</i> — run a command at low priority (sh only)	ConvexOS	NICE(1)
<i>nice</i> — set program priority	ConvexOS	NICE(3C)
Nine-Track Magnetic Tape Device Driver	ConvexOS	TA(4)
<i>nlist</i> — get entries from name list	ConvexOS	NLIST(3)
<i>nm</i> — print name list	ConvexOS	NM(1)
<i>nm68</i> — print 680x0 name list	ConvexOS	NM68(1)
<i>nm88</i> — MC88100 print name list	ConvexOS	NM88(1)
<i>nohup</i> — run a command at low priority (sh only)	ConvexOS	NICE(1)
non-local goto	ConvexOS	SETJMP(3)
non-local goto with signal state preservation	ConvexOS	SIGSETJMP(3)
<i>notes</i> — news system	ConvexOS	NOTES(1)
notesfile dump/load programs	ConvexOS	NFDUMP(8)
notesfile networking programs	ConvexOS	NFXMIT(8)
<i>nroff</i> — text formatting	ConvexOS	NROFF(1)
<i>nslookup</i> — query name servers interactively	ConvexOS	NSLOOKUP(1)
<i>ntohl</i> — convert values between host and network byte order	ConvexOS	BYTEORDER(3N)
<i>ntohs</i> — convert values between host and network byte order	ConvexOS	BYTEORDER(3N)
<i>nu</i> — add a new user to the system	ConvexOS	NU(8)
nu defaults database	ConvexOS	NURC(5)
<i>null</i> — data sink	ConvexOS	NULL(4)
<i>NULL</i> — header file contains standard definitions	ConvexOS	STDDEF.H(3)
numeric ID to name filter	ConvexOS	IDTONAME(1)
<i>/etc/nurc</i> — nu defaults database	ConvexOS	NURC(5)
I/O statistics	ConvexOS	IOSTATS(4)
<i>o68</i> — 680x0 peephole optimizer	ConvexOS	O68(1)
octal, decimal, hex, ascii dump	ConvexOS	OD(1)
<i>od</i> — octal, decimal, hex, ascii dump	ConvexOS	OD(1)
<i>off</i> — enable/disable logins on a tty	ConvexOS	ON(8)
off line print	ConvexOS	LPR(1)

Description	Product	Man Page
<i>offsetof</i> — header file contains standard definitions	ConvexOS	STDDEF.H(3)
<i>on</i> — enable/disable logins on a tty	ConvexOS	ON(8)
online information system	ConvexOS	HELP(1)
<i>op</i> — operator interface tool for giving restricted access to privileged commands	ConvexOS	OP(8)
<i>op.access</i> — list of operator mnemonics and restrictions enforced by the op program	ConvexOS	OP.ACCESS(5)
<i>open</i> — open a file for reading or writing, or create a new file	ConvexOS	OPEN(2)
open a file for reading or writing via a file handle	ConvexOS	FHOPEN(2)
open a file for reading or writing, or create a new file	ConvexOS	OPEN(2)
open a stream	ConvexOS	FOPEN(3S)
open share data base file	ConvexOS	OPENSHARES(3)
<i>opendir</i> — directory operations	ConvexOS	DIRECTORY(3)
<i>openlog</i> — control system log	ConvexOS	SYSLOG(3)
<i>openshares</i> — open share data base file	ConvexOS	OPENSHARES(3)
operator interface tool for giving restricted access to privileged commands	ConvexOS	OP(8)
Operator Request handler	ConvexOS	OPREQ(1)
<i>opreq</i> — Operator Request handler	ConvexOS	OPREQ(1)
opreq daemon	ConvexOS	OPREQ_DAEMON(8)
<i>opreq_daemon</i> — opreq daemon	ConvexOS	OPREQ_DAEMON(8)
<i>opthdr</i> — optional (secondary) header for standard format object files	ConvexOS	OPTHDR(5)
optional (secondary) header for standard format object files	ConvexOS	OPTHDR(5)
output conversion	ConvexOS	ECVT(3)
output the last part of a file	ConvexOS	TAIL(1)
<i>pa</i> — line printer driver	ConvexOS	PA(4)
<i>pac</i> — printer/plotter accounting information	ConvexOS	PAC(8)
<i>page</i> — file perusal filter for CRT viewing	ConvexOS	MORE(1)
<i>pagesize</i> — print system page size	ConvexOS	PAGESIZE(1)
paging device	ConvexOS	DRUM(4)
<i>passwd</i> — change login password	ConvexOS	PASSWD(1)
<i>passwd</i> — password file	ConvexOS	PASSWD(5)
password file	ConvexOS	PASSWD(5)
password restrictions file	ConvexOS	PWRESTRICT(5)
<i>patch</i> — program for applying a diff file to an original	ConvexOS	PATCH(1)
<i>pathconf</i> — configurable pathname variables	ConvexOS	PATHCONF(3)
<i>pattach</i> — process attach	ConvexOS	PATTACH(2)
pattern scanning and processing language	ConvexOS	AWK(1)
<i>pause</i> — stop until signal	ConvexOS	PAUSE(3C)
<i>pax</i> — portable archive exchange	ConvexOS	PAX(1)
<i>pb</i> — Ikon controller for Versatec plotters	ConvexOS	PB(4)
<i>pclose</i> — initiate I/O to/from a process or pipe	ConvexOS	POPEN(3)
periodic accounting utilities	ConvexOS	ACCOUNTING(8)
periodically update the super block	ConvexOS	UPDATE(8)
<i>perl</i> — Practical Extraction and Report Language	ConvexOS	PERL(1)
permit or deny messages	ConvexOS	MESG(1)

Description	Product	Man Page
permuted index	ConvexOS	PTX(1)
<i>perrot</i> — system error messages	ConvexOS	PERROR(3)
<i>pgetregid</i> — get a process' real and effective group ID	ConvexOS	PGETREGID(2)
<i>phones</i> — remote host phone number data base	ConvexOS	PHONES(5)
<i>pi</i> — process interface	ConvexOS	PI(4)
<i>ping</i> — send ICMP ECHO_REQUEST packets to network hosts	ConvexOS	PING(8C)
<i>pipe</i> — create an interprocess communication channel	ConvexOS	PIPE(2)
pipe fitting	ConvexOS	TEE(1)
<i>pl</i> — print share information for designated users	ConvexOS	PL(1)
<i>plot</i> — graphics interface	ConvexOS	PLOT(5)
<i>plot.openpl</i> — graphics interface	ConvexOS	PLOT(3X)
printer/plotter accounting information	ConvexOS	PAC(8)
<i>point</i> — graphics interface	ConvexOS	PLOT(3X)
poll a remote UUCP site	ConvexOS	UUPOLL(8C)
<i>popen</i> — initiate I/O to/from a process or pipe	ConvexOS	POPEN(3)
portable archive exchange	ConvexOS	PAX(1)
POSIX compatible extended cpio archive file format	ConvexOS	CPIO(5)
POSIX compatible extended tape archive file format	ConvexOS	TAR(5)
<i>pow</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>powf</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>pr</i> — print file	ConvexOS	PR(1)
<i>pr</i> to the line printer	ConvexOS	PRINT(1)
<i>pr68</i> — display formatted dump of a 680x0 object file	ConvexOS	PR68(1)
Practical Extraction and Report Language	ConvexOS	PERL(1)
<i>preen</i> — run fsck in parallel over normally mounted disks	ConvexOS	PREEN(8)
prepare execution profile	ConvexOS	MONITOR(3)
primitive system data types	ConvexOS	TYPES(5)
print "C" files	ConvexOS	CPR(1)
<i>print</i> — pr to the line printer	ConvexOS	PRINT(1)
print 680x0 name list	ConvexOS	NM68(1)
print and set the date	ConvexOS	DATE(1)
print calendar	ConvexOS	CAL(1)
print file	ConvexOS	PR(1)
print information from the kernel's stripe table	ConvexOS	GETST(8)
print log messages and other information about RCS files	ConvexOS	RLOG(1)
print name list	ConvexOS	NM(1)
print number of notes with/without director's messages in a notesfile	ConvexOS	NFCOUNT(1)
print out mail in the post office	ConvexOS	PRMAIL(1)
print out system information	ConvexOS	GETSYSINFO(1)
print out the environment	ConvexOS	PRINTENV(1)
print selected share data base file entries using printf formats	ConvexOS	PWINTF(1)
print share information for designated users	ConvexOS	PL(1)
print statistics about Notesfiles	ConvexOS	NFSTATS(1)
print system facts	ConvexOS	PSTAT(8)

Description	Product	Man Page
print system page size	ConvexOS	PAGESIZE(1)
print tape request queue and tape unit utilization	ConvexOS	TPQ(1)
print tape request queue and tape unit utilization	ConvexOS	TPQUEUE(1)
print the contents of a notesfile	ConvexOS	NFPRINT(1)
print the effective or real current user ID	ConvexOS	WHOAMI(1)
print the queue of jobs waiting to be run	ConvexOS	ATQ(1)
print wordy sentences; thesaurus for diction	ConvexOS	DICTION(1)
print wordy sentences; thesaurus for diction	ConvexOS	EXPLAIN(1)
<i>printcap</i> — printer capability database	ConvexOS	PRINTCAP(5)
<i>printenv</i> — print out the environment	ConvexOS	PRINTENV(1)
printer accounting file	ConvexOS	LPD-ACCT(5)
printer capability database	ConvexOS	PRINTCAP(5)
<i>printf</i> — formatted output conversion	ConvexOS	PRINTF(3S)
<i>prmail</i> — print out mail in the post office	ConvexOS	PRMAIL(1)
process attach	ConvexOS	PATTACH(2)
process checkpoint file format	ConvexOS	CHKPNT(5)
process control initialization	ConvexOS	INIT(8)
process interface	ConvexOS	PI(4)
process nroff-style tty37 output for Printronix printers	ConvexOS	LPRMAN(1)
process status	ConvexOS	PS(1)
process tape archives	ConvexOS	TAR(1)
<i>profil</i> — execution time profile	ConvexOS	PROFIL(2)
program for applying a diff file to an original	ConvexOS	PATCH(1)
program verification	ConvexOS	ASSERT(3X)
protocol name data base	ConvexOS	PROTOCOLS(5)
<i>protocols</i> — protocol name data base	ConvexOS	PROTOCOLS(5)
provide truth values	ConvexOS	FALSE(1)
provide truth values	ConvexOS	TRUE(1)
<i>prx</i> — filter for Printronix printers	ConvexOS	PRX(1)
<i>ps</i> — process status	ConvexOS	PS(1)
<i>psetregid</i> — set a process' real and effective group ID	ConvexOS	PSETREGID(2)
pseudo terminal driver	ConvexOS	PTY(4)
<i>psignal</i> — system signal messages	ConvexOS	PSIGNAL(3)
<i>pstat</i> — print system facts	ConvexOS	PSTAT(8)
<i>ptr_diff_t</i> — header file contains standard definitions	ConvexOS	STDDEF.H(3)
<i>ptr</i> — permuted index	ConvexOS	PTX(1)
<i>pty</i> — pseudo terminal driver	ConvexOS	PTY(4)
push character back into input stream	ConvexOS	UNGETC(3S)
put a string on a stream	ConvexOS	PUTS(3S)
put character or word on a stream	ConvexOS	PUTC(3S)
<i>putc</i> — put character or word on a stream	ConvexOS	PUTC(3S)
<i>putchar</i> — put character or word on a stream	ConvexOS	PUTC(3S)
<i>puts</i> — put a string on a stream	ConvexOS	PUTS(3S)
<i>putshares</i> — write share data base file entry	ConvexOS	PUTSHARES(3)
<i>putst</i> — manipulate kernel databases for striped disk partitions	ConvexOS	PUTST(8)
<i>putw</i> — put character or word on a stream	ConvexOS	PUTC(3S)
<i>pwd</i> — working directory name	ConvexOS	PWD(1)
<i>pwintf</i> — print selected share data base file entries using printf formats	ConvexOS	PWINTF(1)

Description	Product	Man Page
<i>pwrestrict</i> — password restrictions file	ConvexOS	PWRESTRICT(5)
<i>qsort</i> — quicker sort and search	ConvexOS	QSORT(3)
query name servers interactively	ConvexOS	NSLOOKUP(1)
quicker sort and search	ConvexOS	QSORT(3)
<i>quot</i> — summarize file system ownership	ConvexOS	QUOT(8)
<i>quota</i> — display disk usage and limits	ConvexOS	QUOTA(1)
<i>quotacheck</i> — check file system quota consistency	ConvexOS	QUOTACHECK(8)
<i>quotactl</i> — manipulate disk quotas	ConvexOS	QUOTACTL(2)
<i>quotaoff</i> — turn file system quotas on and off	ConvexOS	QUOTAON(8)
<i>quotaon</i> — turn file system quotas on and off	ConvexOS	QUOTAON(8)
<i>raise</i> — simplified software signal facilities	ConvexOS	SIGNAL(3C)
<i>rand</i> — random number generator	ConvexOS	RAND(3C)
<i>random</i> — better random number generator; routines for changing generators	ConvexOS	RANDOM(3)
random number generator	ConvexOS	RAND(3C)
<i>ranlib</i> — convert archives to random libraries	ConvexOS	RANLIB(1)
Raster Technologies Model One/80 driver	ConvexOS	DM(4)
<i>rates</i> — show system scheduling rates by scheduling group	ConvexOS	RATES(1)
<i>rc</i> — command script for auto-reboot and daemons	ConvexOS	RC(8)
<i>rcmd</i> — routines for returning a stream to a remote command	ConvexOS	RCMD(3X)
<i>rcs</i> — change RCS file attributes	ConvexOS	RCS(1)
<i>rcsdiff</i> — compare RCS revisions	ConvexOS	RCSDIFF(1)
<i>rcsfile</i> — format of RCS file	ConvexOS	RCSFILE(5)
<i>rcsmerge</i> — merge RCS revisions	ConvexOS	RCSMERGE(1)
<i>rcvtir</i> — IEEE/native floating point mode conversion routines.	ConvexOS	RCVTIR(3M)
<i>rdist</i> — remote file distribution program	ConvexOS	RDIST(1)
<i>rdump</i> — dump file systems across the network	ConvexOS	RDUMP(8C)
<i>re_comp</i> — regular expression handler	ConvexOS	REGEX(3)
<i>re_exec</i> — regular expression handler	ConvexOS	REGEX(3)
<i>read</i> — read from a file	ConvexOS	READ(2)
read a password	ConvexOS	GETPASS(3)
read from a file	ConvexOS	READ(2)
read or write ANSI multifile labeled tapes	ConvexOS	ANSITAR(1)
read physical memory from a crash dump tape	ConvexOS	CRASHREAD(8)
read scattered data	ConvexOS	READV(2)
read value of a symbolic link	ConvexOS	READLINK(2)
read, modify, and write shares file entry	ConvexOS	GETSHPUT(3)
<i>readdir</i> — directory operations	ConvexOS	DIRECTORY(3)
<i>readlink</i> — read value of a symbolic link	ConvexOS	READLINK(2)
<i>readnotes</i> — news system	ConvexOS	NOTES(1)
<i>readv</i> — read scattered data	ConvexOS	READV(2)
<i>realloc</i> — memory allocator	ConvexOS	MALLOC(3)
rearrange name list	ConvexOS	SYMORDER(1)
<i>reboot</i> — reboot system or halt processor	ConvexOS	REBOOT(2)
<i>reboot</i> — UNIX bootstrapping procedures	ConvexOS	REBOOT(8)
reboot system or halt processor	ConvexOS	REBOOT(2)

Description	Product	Man Page
rebuild the data base for the mail aliases file	ConvexOS	NEWALIASES(1)
rebuild the whatis database	ConvexOS	MAKEWHATIS(8)
receive a message from a socket	ConvexOS	RECV(2)
<i>recv</i> — receive a message from a socket	ConvexOS	RECV(2)
<i>recvfrom</i> — receive a message from a socket	ConvexOS	RECV(2)
<i>recvmsg</i> — receive a message from a socket	ConvexOS	RECV(2)
regular expression handler	ConvexOS	REGEX(3)
relational database operator	ConvexOS	JOIN(1)
remind you when you have to leave	ConvexOS	LEAVE(1)
reminder service	ConvexOS	CALENDAR(1)
<i>remote</i> — remote host description file	ConvexOS	REMOTE(5)
remote execution server	ConvexOS	REXECD(8C)
remote file distribution program	ConvexOS	RDIST(1)
remote host description file	ConvexOS	REMOTE(5)
remote host phone number data base	ConvexOS	PHONES(5)
remote login server	ConvexOS	RLOGIND(8C)
remote magtape protocol module	ConvexOS	RMT(8C)
remote shell server	ConvexOS	RSHD(8C)
remote user information server	ConvexOS	FINGERD(8C)
remove (unlink) files or directories	ConvexOS	RM(1)
<i>remove</i> — remove a file or directory entry	ConvexOS	UNLINK(2)
remove a directory file	ConvexOS	RMDIR(2)
remove a file or directory entry	ConvexOS	UNLINK(2)
remove a file system	ConvexOS	UNMOUNT(2)
remove an entry from the shares data-base	ConvexOS	REMSHENT(8)
remove text/data/bss sections from an executable	ConvexOS	GUT(1)
remove columns from a file	ConvexOS	COLRM(1)
remove debugger stabs or symbols and relocation bits	ConvexOS	STRIP(1)
remove debugger stabs or symbols and relocation bits from MC88100 object files	ConvexOS	STRIP88(1)
insert/remove element from a queue	ConvexOS	INSQUE(3)
remove jobs from the line printer spooling queue	ConvexOS	LPRM(1)
remove jobs from the tape mount request queue	ConvexOS	TPRM(1)
remove jobs spooled by at	ConvexOS	ATRM(1)
remove labels from a labeled tape	ConvexOS	TPUNLABEL(1)
remove nroff, troff, tbl and eqn constructs	ConvexOS	DEROFF(1)
<i>remque</i> — insert/remove element from a queue	ConvexOS	INSQUE(3)
<i>remshent</i> — remove an entry from the shares data-base	ConvexOS	REMSHENT(8)
<i>rename</i> — change the name of a file	ConvexOS	RENAME(2)
<i>renice</i> — alter priority of running processes	ConvexOS	RENICE(8)
report repeated lines in a file	ConvexOS	UNIQ(1)
report system, disk, TTY, and network activity in windows	ConvexOS	SYSPIC(8)
report virtual memory statistics	ConvexOS	VMSTAT(1)
reposition a stream	ConvexOS	FSEEK(3S)
<i>repquota</i> — summarize quotas for a file system	ConvexOS	REPQUOTA(8)
request tape mount or allocate a drive	ConvexOS	TPMOUNT(1)
<i>res_init</i> — resolver routines	ConvexOS	RESOLVER(3)
<i>res_mkquery</i> — resolver routines	ConvexOS	RESOLVER(3)

Description	Product	Man Page
<i>res_send</i> — resolver routines	ConvexOS	RESOLVER(3)
reserve a terminal	ConvexOS	LOCK(1)
<i>resolver</i> — resolver configuration file	ConvexOS	RESOLVER(5)
resolver configuration file	ConvexOS	RESOLVER(5)
resolver routines	ConvexOS	RESOLVER(3)
<i>restart</i> — restart execution of a process or a process family	ConvexOS	RESTART(1)
<i>restart</i> — restart execution of a process or a process family	ConvexOS	RESTART(3)
restart execution of a process or a process family	ConvexOS	RESTART(1)
restart execution of a process or a process family	ConvexOS	RESTART(3)
<i>restore</i> — incremental file system restore	ConvexOS	RESTORE(8)
restore a file system dump across the network	ConvexOS	RRESTORE(8C)
return a path to a file from a file descriptor	ConvexOS	FDPATH(2)
return or set limits or cost structures	ConvexOS	LIMITS(2)
return stream to a remote command	ConvexOS	REXEC(3X)
return the path a file was opened with	ConvexOS	FHPATH(2)
<i>rev</i> — reverse lines of a file	ConvexOS	REV(1)
reverse lines of a file	ConvexOS	REV(1)
<i>rew</i> — line printer rewind commands	ConvexOS	LPREWIND(8)
<i>rewind</i> — reposition a stream	ConvexOS	FSEEK(3S)
<i>rewinddir</i> — directory operations	ConvexOS	DIRECTORY(3)
<i>rexec</i> — return stream to a remote command	ConvexOS	REXEC(3X)
<i>rexecd</i> — remote execution server	ConvexOS	REXECD(8C)
<i>rhosts</i> — list of trusted hosts, remote host access file	ConvexOS	HOSTS.EQUIV(5)
<i>rindex</i> — string operations	ConvexOS	STRING(3)
<i>rindex</i> — string search functions	ConvexOS	STRINGSEARCH(3)
<i>rivet</i> — Rivet a symbol table to the end of a vmunix file	ConvexOS	RIVET(8)
Rivet a symbol table to the end of a vmunix file	ConvexOS	RIVET(8)
<i>rlog</i> — print log messages and other information about RCS files	ConvexOS	RLOG(1)
<i>rlogind</i> — remote login server	ConvexOS	RLOGIND(8C)
<i>rm</i> — remove (unlink) files or directories	ConvexOS	RM(1)
<i>rmail</i> — handle remote mail received via uucp	ConvexOS	RMAIL(1)
<i>rmdir</i> — remove (unlink) files or directories	ConvexOS	RM(1)
<i>rmdir</i> — remove a directory file	ConvexOS	RMDIR(2)
<i>rmnf</i> — create and delete notesfiles	ConvexOS	MKNF(8)
<i>rmt</i> — remote magtape protocol module	ConvexOS	RMT(8C)
<i>route</i> — manually manipulate the routing tables	ConvexOS	ROUTE(8C)
<i>routed</i> — network routing daemon	ConvexOS	ROUTED(8C)
routines for returning a stream to a remote command	ConvexOS	RCMD(3X)
<i>rpc.lockd</i> — network lock daemon	ConvexOS	LOCKD(8C)
<i>rpc.statd</i> — network status monitor	ConvexOS	STATD(8C)
<i>rrestore</i> — restore a file system dump across the network	ConvexOS	RRESTORE(8C)
<i>rreport</i> — routines for returning a stream to a remote command	ConvexOS	RCMD(3X)
<i>rshd</i> — remote shell server	ConvexOS	RSHD(8C)

Description	Product	Man Page
run a command at low priority (sh only)	ConvexOS	NICE(1)
run fsck in parallel over normally mounted disks	ConvexOS	PREEN(8)
<i>ruserok</i> — routines for returning a stream to a remote command	ConvexOS	RCMD(3X)
<i>ruhod</i> — system status server	ConvexOS	RWHOD(8C)
<i>sa</i> — system accounting	ConvexOS	SA(8)
<i>sacos</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>sasin</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>satan</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>satan2</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>sbrk</i> — change data segment size	ConvexOS	BRK(2)
<i>scabs</i> — Euclidean distance	ConvexOS	HYPOT(3M)
scan a directory	ConvexOS	SCANDIR(3)
<i>scandir</i> — scan a directory	ConvexOS	SCANDIR(3)
<i>scanf</i> — formatted input conversion	ConvexOS	SCANF(3S)
<i>sccstorcs</i> — build RCS file from SCCS file	ConvexOS	SCCSTORCS(1)
schedule signal after specified time	ConvexOS	ALARM(3C)
<i>scnhdr</i> — section header for standard format object files	ConvexOS	SCNHDR(5)
<i>scos</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>scosh</i> — hyperbolic functions	ConvexOS	SINH(3M)
screen functions with "optimal" cursor motion	ConvexOS	CURSES(3X)
screen oriented (visual) text editors based on <i>ex</i>	ConvexOS	VI(1)
<i>script</i> — make typescript of terminal session	ConvexOS	SCRIPT(1)
search a file for a pattern	ConvexOS	GREP(1)
section header for standard format object files	ConvexOS	SCNHDR(5)
<i>sed</i> — stream editor (non-interactive text editor)	ConvexOS	SED(1)
<i>seestat</i> — gather system statistics and print graphs of system statistics	ConvexOS	SEESTAT(8)
<i>select</i> — synchronous I/O multiplexing	ConvexOS	SELECT(2)
select or reject lines common to two sorted files	ConvexOS	COMM(1)
<i>send</i> — initiate a UUCP phone call to a neighboring site	ConvexOS	SEND(1)
<i>send</i> — send a message to a socket	ConvexOS	SEND(2)
send a message to a socket	ConvexOS	SEND(2)
send and receive mail	ConvexOS	MAIL(1)
send ICMP ECHO_REQUEST packets to network hosts	ConvexOS	PING(8C)
send mail over the internet	ConvexOS	SENDMAIL(8)
send or receive mail among users	ConvexOS	BINMAIL(1)
send signal to a process	ConvexOS	KILL(2)
send signal to a process group	ConvexOS	KILLPG(2)
<i>sendmail</i> — send mail over the internet	ConvexOS	SENDMAIL(8)
<i>sendmsg</i> — send a message to a socket	ConvexOS	SEND(2)
<i>sendto</i> — send a message to a socket	ConvexOS	SEND(2)
service name data base	ConvexOS	SERVICES(5)
<i>services</i> — service name data base	ConvexOS	SERVICES(5)
set a process' activity ID	ConvexOS	SETAID(2)
set a process' real and effective group ID	ConvexOS	PSETREGID(2)

Description	Product	Man Page
set and get terminal state (defunct)	ConvexOS	STTY(3C)
set current signal mask	ConvexOS	SIGSETPMASK(2)
get/set date and time	ConvexOS	GETTIMEOFDAY(2)
set file creation mode mask	ConvexOS	UMASK(2)
set file times	ConvexOS	UTIME(3C)
set file times	ConvexOS	UTIMES(2)
set floating point mode during program execution	ConvexOS	SETFFPMODE(3)
set group access list	ConvexOS	SETGROUPS(2)
set kernel shares for a user	ConvexOS	SETUPSHARES(3)
set limits structure	ConvexOS	SETLIMITS(3)
get/set name of current domain	ConvexOS	GETDOMAINNAME(2)
get/set name of current host	ConvexOS	GETHOSTNAME(2)
set and/or get signal stack context	ConvexOS	SIGSTACK(2)
set or print identifier of current host system	ConvexOS	HOSTID(1)
set or print name of current host system	ConvexOS	HOSTNAME(1)
get/set process attributes	ConvexOS	GETPATTR(2)
get/set process entry flags	ConvexOS	GETPFLAGS(2)
set process group	ConvexOS	SETPGID(2)
set process group	ConvexOS	SETPGRP(2)
set process id	ConvexOS	SETPID(2)
set program priority	ConvexOS	NICE(3C)
get/set program scheduling priority	ConvexOS	GETPRIORITY(2)
set real and effective group ID	ConvexOS	SETREGID(2)
set real and effective user ID's	ConvexOS	SETREUID(2)
get/set terminal characteristics	ConvexOS	TCGETATTR(3)
get/set terminal input/output speed	ConvexOS	CFGETOSPEED(3)
set terminal mode	ConvexOS	GETTY(8)
set terminal options	ConvexOS	STTY(1)
get/set terminal process group	ConvexOS	TCGETPGRP(3)
set terminal tabs	ConvexOS	TABS(1)
set the HYPERchannel routing tables	ConvexOS	HYROUTE(8C)
Set the number of HASP Host communications boards	ConvexOS	SETHHB(8)
set time zone information	ConvexOS	TZSET(3)
get/set unique identifier of current host	ConvexOS	GETHOSTID(2)
set up another user's shares and run a (shell) command	ConvexOS	SL(1)
set user and group ID	ConvexOS	SETUID(3)
get/set value of interval timer	ConvexOS	GETITIMER(2)
<i>setactent</i> — get activity file entry	ConvexOS	GETACTENT(3)
<i>setacwent</i> — get actwho file entry	ConvexOS	GETACWENT(3)
<i>setaid</i> — set a process' activity ID	ConvexOS	SETAID(2)
<i>setbuf</i> — assign buffering to a stream	ConvexOS	SETBUF(3S)
<i>setbuffer</i> — assign buffering to a stream	ConvexOS	SETBUF(3S)
<i>setdomainname</i> — get/set name of current domain	ConvexOS	GETDOMAINNAME(2)
<i>setenv</i> — manipulate environmental variables	ConvexOS	GETENV(3)
<i>setfpmode</i> — set floating point mode during program execution	ConvexOS	SETFFPMODE(3)
<i>setfsent</i> — get file system descriptor file entry	ConvexOS	GETFSSENT(3X)
<i>setgid</i> — set user and group ID	ConvexOS	SETUID(3)
<i>setgrent</i> — get group file entry	ConvexOS	GETGRENT(3)

Description	Product	Man Page
<i>setgroups</i> — set group access list	ConvexOS	SETGROUPS(2)
<i>sethbb</i> — Set the number of HASP Host communications boards	ConvexOS	SETHBB(8)
<i>sethostent</i> — get network host entry	ConvexOS	GETHOSTBYNAME(3N)
<i>sethostid</i> — get/set unique identifier of current host	ConvexOS	GETHOSTID(2)
<i>sethostname</i> — get/set name of current host	ConvexOS	GETHOSTNAME(2)
<i>setitimer</i> — get/set value of interval timer	ConvexOS	GETITIMER(2)
<i>setjmp</i> — non-local goto	ConvexOS	SETJMP(3)
<i>setkey</i> — encryption operations	ConvexOS	CRYPT(3)
<i>setlimits</i> — set limits structure	ConvexOS	SETLIMITS(3)
<i>setlinebuf</i> — assign buffering to a stream	ConvexOS	SETBUF(3S)
<i>setlocale</i> — functions for locale manipulation	ConvexOS	SETLOCALE(3)
<i>setlogmask</i> — control system log	ConvexOS	SYSLOG(3)
<i>setmntent</i> — get file system descriptor file entry	ConvexOS	GETMNTENT(3)
<i>setnetent</i> — get network entry	ConvexOS	GETNETENT(3N)
<i>setpatrr</i> — get/set process attributes	ConvexOS	GETPATRR(2)
<i>setpflags</i> — get/set process entry flags	ConvexOS	GETPFLAGS(2)
<i>setpgid</i> — set process group	ConvexOS	SETPGID(2)
<i>setpgrp</i> — set process group	ConvexOS	SETPGRP(2)
<i>setpid</i> — set process id	ConvexOS	SETPID(2)
<i>setpriority</i> — get/set program scheduling priority	ConvexOS	GETPRIORITY(2)
<i>setprotoent</i> — get protocol entry	ConvexOS	GETPROTOENT(3N)
<i>setpwent</i> — get password file entry	ConvexOS	GETPWENT(3)
<i>setpwrestent</i> — get entry from password restrictions file	ConvexOS	GETPWRESTENT(3)
<i>setregid</i> — set real and effective group ID	ConvexOS	SETREGID(2)
<i>setreuid</i> — set real and effective user ID's	ConvexOS	SETREUID(2)
<i>setrlimit</i> — control maximum system resource consumption	ConvexOS	GETRLIMIT(2)
<i>setservent</i> — get service entry	ConvexOS	GETSERVENT(3N)
<i>setsid</i> — create session and set process group ID	ConvexOS	SETSID(2)
<i>setsockopt</i> — get and set options on sockets	ConvexOS	GETSOCKOPT(2)
<i>setstate</i> — better random number generator; routines for changing generators	ConvexOS	RANDOM(3)
<i>settimeofday</i> — get/set date and time	ConvexOS	GETTIMEOFDAY(2)
<i>setttyent</i> — get ttys file entry	ConvexOS	GETTTYENT(3)
<i>setuid</i> — set user and group ID	ConvexOS	SETUID(3)
<i>setupshares</i> — set kernel shares for a user	ConvexOS	SETUPSHARES(3)
<i>setusershell</i> — get legal user shells	ConvexOS	GETUSERSHELL(3)
<i>setvbuf</i> — assign buffering to a stream	ConvexOS	SETBUF(3S)
<i>sezp</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>sfabs</i> — absolute value, floor, ceiling functions	ConvexOS	FLOOR(3M)
<i>sh</i> — command language	ConvexOS	SH(1)
<i>Share</i> — Share Scheduling on CONVEX Unix	ConvexOS	SHARE(5)
Share Scheduling on CONVEX Unix	ConvexOS	SHARE(5)
<i>sharecf</i> — configure and display a Share configuration	ConvexOS	SHARECF(8)
shared memory synchronization primitives	ConvexOS	MSET(3)
shared memory synchronization primitives	ConvexOS	MSLEEP(2)
<i>sharer</i> — clean up defunct user Inode structures	ConvexOS	SHARER(8)

Description	Product	Man Page
<i>/etc/shares</i> — shares data-base file for Share system	ConvexOS	SHARES(5)
shares data-base file for Share system	ConvexOS	SHARES(5)
<i>sharesfile</i> — change name of shares file	ConvexOS	SHARESFILE(3)
<i>sharesh</i> — configure and display a Share configuration	ConvexOS	SHARECF(8)
shell (command interpreter) with C-like syntax	ConvexOS	CSH(1)
<i>/etc/shells</i> — define the standard user shells	ConvexOS	SHELLS(5)
show charges for share system usage calculations	ConvexOS	CHARGES(1)
show group memberships	ConvexOS	GROUPS(1)
show how long system has been up	ConvexOS	UPTIME(1)
show last commands executed in reverse order	ConvexOS	LASTCOMM(1)
show snapshot of the UUCP system	ConvexOS	UUSNAP(8C)
show system scheduling rates by scheduling group	ConvexOS	RATES(1)
show what versions of object modules were used to construct a file	ConvexOS	WHAT(1)
show yesterday's date	ConvexOS	YESTERDAY(1)
shut down part of a full-duplex connection	ConvexOS	SHUTDOWN(2)
<i>shutdown</i> — close down the system at a given time	ConvexOS	SHUTDOWN(8)
<i>shutdown</i> — shut down part of a full-duplex connection	ConvexOS	SHUTDOWN(2)
<i>shypot</i> — Euclidean distance	ConvexOS	HYPOT(3M)
<i>sigaction</i> — examine and change signal action	ConvexOS	SIGACTION(2)
<i>sigaddset</i> — manipulate signal sets	ConvexOS	SIGSETTOPS(3)
<i>sigblock</i> — block or unblock signals	ConvexOS	SIGBLOCK(2)
<i>sigdelset</i> — manipulate signal sets	ConvexOS	SIGSETTOPS(3)
<i>sigemptyset</i> — manipulate signal sets	ConvexOS	SIGSETTOPS(3)
<i>sigfillset</i> — manipulate signal sets	ConvexOS	SIGSETTOPS(3)
<i>sigismember</i> — manipulate signal sets	ConvexOS	SIGSETTOPS(3)
<i>siglongjmp</i> — non-local goto with signal state preservation	ConvexOS	SIGSETJMP(3)
sign on	ConvexOS	LOGIN(1)
<i>signal</i> — simplified software signal facilities	ConvexOS	SIGNAL(3C)
<i>sigpause</i> — atomically release blocked signals and wait for interrupt	ConvexOS	SIGPAUSE(2)
<i>sigpending</i> — examine pending signals	ConvexOS	SIGPENDING(2)
<i>sigprocmask</i> — block or unblock signals	ConvexOS	SIGPROCMAK(3)
<i>sigsetjmp</i> — non-local goto with signal state preservation	ConvexOS	SIGSETJMP(3)
<i>sigsetmask</i> — set current signal mask	ConvexOS	SIGSETMASK(2)
<i>sigstack</i> — set and/or get signal stack context	ConvexOS	SIGSTACK(2)
<i>sigsuspend</i> — wait for signal	ConvexOS	SIGSUSPEND(3)
<i>sigunblock</i> — block or unblock signals	ConvexOS	SIGBLOCK(2)
<i>sigvec</i> — software signal facilities	ConvexOS	SIGVEC(2)
simple text formatter	ConvexOS	FMT(1)
simplified software signal facilities	ConvexOS	SIGNAL(3C)
<i>sin</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>sinf</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>sinh</i> — hyperbolic functions	ConvexOS	SINH(3M)
<i>sinhf</i> — hyperbolic functions	ConvexOS	SINH(3M)
<i>size</i> — size of an object file	ConvexOS	SIZE(1)
<i>size</i> of a 680x0 object file	ConvexOS	SIZE68(1)

Description	Product	Man Page
size of an MC88100 object file	ConvexOS	SIZE88(1)
size of an object file	ConvexOS	SIZE(1)
<i>size68</i> — size of a 680x0 object file	ConvexOS	SIZE68(1)
<i>size88</i> — size of an MC88100 object file	ConvexOS	SIZE88(1)
<i>size_t</i> — header file contains standard definitions	ConvexOS	STDDEF.H(3)
<i>sl</i> — set up another user's shares and run a (shell) command	ConvexOS	SL(1)
<i>slattach</i> — attach serial lines as network interfaces	ConvexOS	SLATTACH(8C)
<i>sleep</i> — suspend execution for an interval	ConvexOS	SLEEP(1)
<i>sleep</i> — suspend execution for interval	ConvexOS	SLEEP(3)
<i>slog</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>slog10</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>slqt</i> — set up another user's shares and run a (shell) command	ConvexOS	SL(1)
<i>sniff</i> — continually watch the end of a file	ConvexOS	SNIFF(1)
<i>socket</i> — create an endpoint for communication	ConvexOS	SOCKET(2)
<i>socketpair</i> — create a pair of connected sockets	ConvexOS	SOCKETPAIR(2)
<i>sod</i> — standard object file dump utility	ConvexOS	SOD(1)
<i>soelim</i> — eliminate .so's from nroff input	ConvexOS	SOELIM(1)
software loopback network interface	ConvexOS	LO(4)
software signal facilities	ConvexOS	SIGVEC(2)
<i>sort</i> — sort or merge files	ConvexOS	SORT(1)
sort or merge files	ConvexOS	SORT(1)
<i>space</i> — graphics interface	ConvexOS	PLOT(3X)
spawn new process in a virtual memory efficient way	ConvexOS	VFORK(2)
special character definitions for eqn	ConvexOS	EQNCHAR(7)
specify additional device for paging and swapping	ConvexOS	SWAPON(8)
specify key bindings for less	ConvexOS	LESSKEY(1)
<i>spell</i> — find spelling errors	ConvexOS	SPELL(1)
<i>spellin</i> — find spelling errors	ConvexOS	SPELL(1)
<i>spellout</i> — find spelling errors	ConvexOS	SPELL(1)
<i>split</i> — split a file into pieces	ConvexOS	SPLIT(1)
split a file into pieces	ConvexOS	SPLIT(1)
split into mantissa and exponent	ConvexOS	FREXP(3)
spool queue examination program	ConvexOS	LPQ(1)
<i>spow</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>sprintf</i> — formatted output conversion	ConvexOS	PRINTF(3S)
<i>spu</i> — transfer files between CONVEX UNIX and SPU UNIX file systems	ConvexOS	SPU(8)
<i>spucmd</i> — execute single SPU UNIX command from CONVEX UNIX	ConvexOS	SPUCMD(8)
<i>sqrt</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>srtf</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>srand</i> — random number generator	ConvexOS	RAND(3C)
<i>srandom</i> — better random number generator; routines for changing generators	ConvexOS	RANDOM(3)
<i>sscanf</i> — formatted input conversion	ConvexOS	SCANF(3S)
<i>ssin</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>ssinh</i> — hyperbolic functions	ConvexOS	SINH(3M)

Description	Product	Man Page
<i>ssqrt</i> — exponential, logarithm, power, square root	ConvexOS	EXP(3M)
<i>st</i> — stripe (disk) device interface	ConvexOS	ST(4)
<i>stab</i> — symbol table types	ConvexOS	STAB(5)
<i>stan</i> — trigonometric functions	ConvexOS	SIN(3M)
standard buffered input/output package	ConvexOS	INTRO(3S)
standard integral functions	ConvexOS	ABS(3)
standard object file dump utility	ConvexOS	SOD(1)
<i>stanh</i> — hyperbolic functions	ConvexOS	SINH(3M)
Start HASP Host	ConvexOS	HHSTART(8)
<i>stat</i> — gather system statistics and print graphs of system statistics	ConvexOS	SEESTAT(8)
<i>stat</i> — get file status	ConvexOS	STAT(2)
<i>statd</i> — network status monitor	ConvexOS	STATD(8C)
statd directory and file structures	ConvexOS	STATMON(5)
<i>state</i> — statd directory and file structures	ConvexOS	STATMON(5)
<i>statfs</i> — get file system statistics	ConvexOS	STATFS(2)
static information about filesystems	ConvexOS	FSTAB(5)
static information about filesystems	ConvexOS	MNTENT(5)
<i>statmon</i> — statd directory and file structures	ConvexOS	STATMON(5)
<i>/usr/adm/stat/stats*</i> — system statistics	ConvexOS	STAT(5)
<i>stdarg</i> — variable argument list	ConvexOS	STDARG.H(3)
stdarg style formatted output conversion	ConvexOS	VPRINTF(3S)
<i>stdarg.h</i> — variable argument list	ConvexOS	STDARG.H(3)
<i>stddef.h</i> — header file contains standard definitions	ConvexOS	STDDEF.H(3)
<i>stdio</i> — standard buffered input/output package	ConvexOS	INTRO(3S)
<i>sticky</i> — file mode sticky bit	ConvexOS	STICKY(8)
Stop HASP Host	ConvexOS	HHSTOP(8)
stop the processor	ConvexOS	HALT(8)
stop until signal	ConvexOS	PAUSE(3C)
<i>store</i> — data base subroutines	ConvexOS	DBM(3X)
<i>strcat</i> — string concatenation functions	ConvexOS	STRINGCAT(3)
<i>strchr</i> — string search functions	ConvexOS	STRINGSEARCH(3)
<i>strcmp</i> — string comparison functions	ConvexOS	STRINGCMP(3)
<i>strcoll</i> — string comparison functions	ConvexOS	STRINGCMP(3)
<i>strcpy</i> — string copy functions	ConvexOS	STRINGCOPY(3)
<i>strcspn</i> — string search functions	ConvexOS	STRINGSEARCH(3)
stream editor (non-interactive text editor)	ConvexOS	SED(1)
stream status inquiries	ConvexOS	FERROR(3S)
<i>strerror</i> — system error messages	ConvexOS	PERROR(3)
<i>strftime</i> — date and time manipulation routines	ConvexOS	CTIME(3)
string comparison functions	ConvexOS	STRINGCMP(3)
string concatenation functions	ConvexOS	STRINGCAT(3)
string copy functions	ConvexOS	STRINGCOPY(3)
string operations	ConvexOS	STRING(3)
string search functions	ConvexOS	STRINGSEARCH(3)
string to numeric value conversion routines	ConvexOS	STRTOD(3)
<i>strings</i> — find the printable strings in a object, or other binary, file	ConvexOS	STRINGS(1)
<i>strip</i> — remove debugger stabs or symbols and relocation bits	ConvexOS	STRIP(1)

Description	Product	Man Page
strip filename affixes	ConvexOS	BASENAME(1)
<i>strip88</i> — remove debugger stabs or symbols and relocation bits from MC88100 object files	ConvexOS	STRIP88(1)
stripe (disk) device interface	ConvexOS	ST(4)
<i>stripecap</i> — striped disk partition description data base	ConvexOS	STRIPECAP(5)
striped disk partition description data base	ConvexOS	STRIPECAP(5)
<i>strlen</i> — string operations	ConvexOS	STRING(3)
<i>strncat</i> — string concatenation functions	ConvexOS	STRINGCAT(3)
<i>strncmp</i> — string comparison functions	ConvexOS	STRINGCMP(3)
<i>strncpy</i> — string copy functions	ConvexOS	STRINGCOPY(3)
<i>strpbrk</i> — string search functions	ConvexOS	STRINGSEARCH(3)
<i>strrchr</i> — string search functions	ConvexOS	STRINGSEARCH(3)
<i>strspn</i> — string search functions	ConvexOS	STRINGSEARCH(3)
<i>strstr</i> — string search functions	ConvexOS	STRINGSEARCH(3)
<i>strtod</i> — string to numeric value conversion routines	ConvexOS	STRTOD(3)
<i>strtok</i> — string search functions	ConvexOS	STRINGSEARCH(3)
<i>strtol</i> — string to numeric value conversion routines	ConvexOS	STRTOD(3)
<i>strtol</i> — string to numeric value conversion routines	ConvexOS	STRTOD(3)
<i>strxfrm</i> — string comparison functions	ConvexOS	STRINGCMP(3)
<i>stty</i> — set and get terminal state (defunct)	ConvexOS	STTY(3C)
<i>stty</i> — set terminal options	ConvexOS	STTY(1)
<i>style</i> — analyze surface characteristics of a document	ConvexOS	STYLE(1)
<i>su</i> — substitute user ID temporarily	ConvexOS	SU(1)
<i>submit</i> — submit batch queue jobs	ConvexOS	SUBMIT(1)
submit a CONVEX problem report	ConvexOS	CONTACT(1)
submit batch queue jobs	ConvexOS	SUBMIT(1)
substitute user ID temporarily	ConvexOS	SU(1)
<i>sum</i> — sum and count blocks in a file	ConvexOS	SUM(1)
sum and count blocks in a file	ConvexOS	SUM(1)
summarize disk usage	ConvexOS	DU(1)
summarize file system ownership	ConvexOS	QUOT(8)
summarize quotas for a file system	ConvexOS	REPUOTA(8)
<i>sumscripts</i> — accounting summary awk scripts	ConvexOS	SUMSCRIPTS(8)
suspend execution for an interval	ConvexOS	SLEEP(1)
suspend execution for interval	ConvexOS	SLEEP(3)
<i>swab</i> — swap bytes	ConvexOS	SWAB(3)
swap bytes	ConvexOS	SWAB(3)
<i>swapon</i> — add a swap device for interleaved paging/swapping	ConvexOS	SWAPON(2)
<i>swapon</i> — specify additional device for paging and swapping	ConvexOS	SWAPON(8)
symbol table types	ConvexOS	STAB(5)
<i>symlink</i> — make symbolic link to a file	ConvexOS	SYMLINK(2)
<i>symorder</i> — rearrange name list	ConvexOS	SYMORDER(1)
<i>sync</i> — update super-block	ConvexOS	SYNC(2)
<i>sync</i> — update the super block	ConvexOS	SYNC(8)
synchronize a file's in-memory state with that on disk	ConvexOS	FSYNC(2)
synchronize shared memory segment with file system	ConvexOS	MSYNC(2)
synchronous I/O multiplexing	ConvexOS	SELECT(2)

Description	Product	Man Page
<i>sys_siglist</i> — system signal messages	ConvexOS	PSIGNAL(3)
<i>syscall</i> — indirect system call	ConvexOS	SYSCALL(2)
<i>sysconf</i> — get configurable system variables	ConvexOS	SYSCONF(3)
<i>syssez</i> — system exerciser that simulates stressful load conditions	ConvexOS	SYSEX(8)
<i>sysgen</i> — tool to aid in building customized operating system images	ConvexOS	SYSGEN(8)
<i>syslog</i> — control system log	ConvexOS	SYSLOG(3)
<i>syslogd</i> — log systems messages	ConvexOS	SYSLOGD(8)
<i>syspic</i> — report system, disk, TTY, and network activity in windows	ConvexOS	SYPIC(8)
<i>system</i> — issue a shell command	ConvexOS	SYSTEM(3)
system accounting	ConvexOS	SA(8)
system configuration file for contact	ConvexOS	CONTACTCAP(5)
system error messages	ConvexOS	PERROR(3)
system exerciser that simulates stressful load conditions	ConvexOS	SYSEX(8)
system messages and junk mail program	ConvexOS	MSG(1)
System I/O configuration description file	ConvexOS	IOCONFIG(4)
system signal messages	ConvexOS	PSIGNAL(3)
system statistics	ConvexOS	STAT(5)
system status server	ConvexOS	RWHOD(8C)
<i>ta</i> — Nine-Track Magnetic Tape Device Driver	ConvexOS	TA(4)
<i>tabs</i> — set terminal tabs	ConvexOS	TABS(1)
<i>tail</i> — output the last part of a file	ConvexOS	TAIL(1)
<i>talk</i> — talk to another user	ConvexOS	TALK(1)
talk daemon	ConvexOS	TALKD(8)
talk to another user	ConvexOS	TALK(1)
<i>talkd</i> — talk daemon	ConvexOS	TALKD(8)
<i>tan</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>tanf</i> — trigonometric functions	ConvexOS	SIN(3M)
<i>tanh</i> — hyperbolic functions	ConvexOS	SINH(3M)
<i>tanhf</i> — hyperbolic functions	ConvexOS	SINH(3M)
tape allocation accounting file	ConvexOS	TP-ACCT(5)
tape daemon	ConvexOS	TPDAEMON(8)
tape drive allocation programs	ConvexOS	TPALLOC(1)
tape mount request programs	ConvexOS	TPMNT(1)
tape subsystem calls	ConvexOS	TAPE(3)
<i>tar</i> — POSIX compatible extended tape archive file format	ConvexOS	TAR(5)
<i>tar</i> — process tape archives	ConvexOS	TAR(1)
<i>tas</i> — indivisibly test and set a memory location	ConvexOS	TAS(3)
<i>tbl</i> — format tables for nroff.	ConvexOS	TBL(1)
<i>tc</i> — cartridge tape driver	ConvexOS	TC(4)
<i>tcdrain</i> — terminal line control functions	ConvexOS	TCSEENDBREAK(3)
<i>tcflow</i> — terminal line control functions	ConvexOS	TCSEENDBREAK(3)
<i>tcflush</i> — terminal line control functions	ConvexOS	TCSEENDBREAK(3)
<i>tcgetattr</i> — get/set terminal characteristics	ConvexOS	TCGETATTR(3)
<i>tcgetpgrp</i> — get/set terminal process group	ConvexOS	TCGETPGRP(3)
<i>tcp</i> — Internet Transmission Control Protocol	ConvexOS	TCP(4P)

Description	Product	Man Page
<i>tcseendbreak</i> — terminal line control functions	ConvexOS	TCSEENDBREAK(3)
<i>tcsetattr</i> — get/set terminal characteristics	ConvexOS	TCGETATTR(3)
<i>tcsetpgrp</i> — get/set terminal process group	ConvexOS	TCGETPGRP(3)
<i>tee</i> — pipe fitting	ConvexOS	TEE(1)
tell cron daemon to update its event list	ConvexOS	TELLCRON(1)
<i>tellcron</i> — tell cron daemon to update its event list	ConvexOS	TELLCRON(1)
<i>telnetd</i> — DARPA TELNET protocol server	ConvexOS	TELNETD(8C)
<i>term</i> — conventional names for terminals	ConvexOS	TERM(7)
<i>termcap</i> — terminal capability data base	ConvexOS	TERMCAP(5)
terminal capability data base	ConvexOS	TERMCAP(5)
terminal configuration data base	ConvexOS	GETTTYTAB(5)
terminal dependent initialization	ConvexOS	TSET(1)
terminal independent operation routines	ConvexOS	TERMCAP(3X)
terminal initialization data	ConvexOS	TTYS(5)
terminal line control functions	ConvexOS	TCSEENDBREAK(3)
terminal multiplexor	ConvexOS	CA(4)
terminate a process	ConvexOS	EXIT(2)
terminate a process after flushing any pending output	ConvexOS	EXIT(3)
terminate a process with extreme prejudice	ConvexOS	KILL(1)
<i>termios</i> — general terminal interface	ConvexOS	TERMIOS(4)
<i>test</i> — condition command	ConvexOS	TEST(1)
text editor	ConvexOS	ED(1)
text editor	ConvexOS	EX(1)
text formatting	ConvexOS	NROFF(1)
text formatting macros	ConvexOS	MS(7)
<i>text68</i> — disassemble a 680x0 object file	ConvexOS	TEXT68(1)
<i>tgetent</i> — terminal independent operation routines	ConvexOS	TERMCAP(3X)
<i>tgetflag</i> — terminal independent operation routines	ConvexOS	TERMCAP(3X)
<i>tgetnum</i> — terminal independent operation routines	ConvexOS	TERMCAP(3X)
<i>tgetstr</i> — terminal independent operation routines	ConvexOS	TERMCAP(3X)
<i>tgoto</i> — terminal independent operation routines	ConvexOS	TERMCAP(3X)
<i>thread_create</i> — create a new thread	ConvexOS	THREAD_CREATE(2)
three-way file merge	ConvexOS	MERGE(1)
<i>time</i> — get system time	ConvexOS	TIME(3C)
<i>time</i> — time a command	ConvexOS	TIME(1)
time a command	ConvexOS	TIME(1)
time server daemon	ConvexOS	TIMED(8)
<i>timed</i> — time server daemon	ConvexOS	TIMED(8)
timed control program	ConvexOS	TIMEDC(8)
<i>timedc</i> — timed control program	ConvexOS	TIMEDC(8)
<i>times</i> — get process times	ConvexOS	TIMES(3C)
<i>timezone</i> — date and time manipulation routines	ConvexOS	CTIME(3)
<i>tmpfile</i> — create a temporary file or generate a unique file name.	ConvexOS	TMPFILE(3S)
<i>tmpnam</i> — create a temporary file or generate a unique file name.	ConvexOS	TMPFILE(3S)
<i>toascii</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>tolower</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
tool to aid in building customized operating system images	ConvexOS	SYSGEN(8)

Description	Product	Man Page
topological sort	ConvexOS	TSORT(1)
<i>touch</i> — update date last modified of a file	ConvexOS	TOUCH(1)
<i>toupper</i> — character testing and mapping macros	ConvexOS	CTYPE(3)
<i>tpalloc</i> — tape drive allocation programs	ConvexOS	TPALLOC(1)
<i>tpattr</i> — change attributes used for future labeled tape files	ConvexOS	TPATTR(1)
<i>tpattr</i> — tape subsystem calls	ConvexOS	TAPE(3)
<i>tpconfig</i> — configure the tape system	ConvexOS	TPCONFIG(8)
<i>tpdaemon</i> — tape daemon	ConvexOS	TPDAEMON(8)
<i>tpdealoc</i> — tape drive allocation programs	ConvexOS	TPALLOC(1)
<i>tperror</i> — tape subsystem calls	ConvexOS	TAPE(3)
<i>tplabel</i> — create a new labeled tape	ConvexOS	TPLABEL(1)
<i>tplabel</i> — tape subsystem calls	ConvexOS	TAPE(3)
<i>tplist</i> — list label information on a labeled tape	ConvexOS	TPLIST(1)
<i>tpmnt</i> — tape mount request programs	ConvexOS	TPMNT(1)
<i>tpmount</i> — request tape mount or allocate a drive	ConvexOS	TPMOUNT(1)
<i>tpmount</i> — tape subsystem calls	ConvexOS	TAPE(3)
<i>tpq</i> — print tape request queue and tape unit utilization	ConvexOS	TPQ(1)
<i>tpqueue</i> — print tape request queue and tape unit utilization	ConvexOS	TPQUEUE(1)
<i>tpqueue</i> — tape subsystem calls	ConvexOS	TAPE(3)
<i>tprm</i> — remove jobs from the tape mount request queue	ConvexOS	TPRM(1)
<i>tpstatus</i> — tape subsystem calls	ConvexOS	TAPE(3)
<i>tpumnt</i> — tape mount request programs	ConvexOS	TPMNT(1)
<i>tpunlabel</i> — remove labels from a labeled tape	ConvexOS	TPUNLABEL(1)
<i>tpunlabel</i> — tape subsystem calls	ConvexOS	TAPE(3)
<i>tpunmount</i> — tape subsystem calls	ConvexOS	TAPE(3)
<i>tpunmount</i> — unmount a tape or deallocate a drive	ConvexOS	TPUNMOUNT(1)
<i>tputs</i> — terminal independent operation routines	ConvexOS	TERMCAP(3X)
<i>tpwait</i> — tape subsystem calls	ConvexOS	TAPE(3)
<i>tpwait</i> — wait for a tpmount to complete	ConvexOS	TPWAIT(1)
<i>tr</i> — translate characters	ConvexOS	TR(1)
Transfer file from a Convex computer to a remote workstation	ConvexOS	HSEND(8)
transfer files between CONVEX UNIX and SPU UNIX file systems	ConvexOS	SPU(8)
transfer files queued by uuep or uux	ConvexOS	UUCICO(8C)
translate characters	ConvexOS	TR(1)
transliterate protocol trace	ConvexOS	TRPT(8C)
trigonometric functions	ConvexOS	SIN(3M)
<i>trpt</i> — transliterate protocol trace	ConvexOS	TRPT(8C)
<i>true</i> — provide truth values	ConvexOS	FALSE(1)
<i>true</i> — provide truth values	ConvexOS	TRUE(1)
<i>truncate</i> — truncate a file to a specified length	ConvexOS	TRUNCATE(2)
truncate a file to a specified length	ConvexOS	TRUNCATE(2)
truncate arbitrary blocks of a file.	ConvexOS	CVXTRUNCATE(2)
<i>tset</i> — terminal dependent initialization	ConvexOS	TSET(1)

Description	Product	Man Page
<i>tsort</i> — topological sort	ConvexOS	TSORT(1)
<i>tty</i> — general terminal interface	ConvexOS	TTY(4)
<i>tty</i> — get terminal name	ConvexOS	TTY(1)
<i>ttyname</i> — find name of a terminal	ConvexOS	TTYNAME(3)
<i>ttys</i> — terminal initialization data	ConvexOS	TTYS(5)
<i>ttyslot</i> — find name of a terminal	ConvexOS	TTYNAME(3)
<i>ttysize</i> — data base of terminal types by port	ConvexOS	TTYTYPE(5)
turn accounting on or off	ConvexOS	ACCT(2)
turn file system quotas on and off	ConvexOS	QUOTAON(8)
<i>types</i> — primitive system data types	ConvexOS	TYPES(5)
typeset mathematics	ConvexOS	NEQN(1)
<i>tzset</i> — set time zone information	ConvexOS	TZSET(3)
<i>ud</i> — user device (for debugging device drivers)	ConvexOS	UD(4)
<i>udp</i> — Internet User Datagram Protocol	ConvexOS	UDP(4P)
<i>ul</i> — do underlining	ConvexOS	UL(1)
<i>umask</i> — set file creation mode mask	ConvexOS	UMASK(2)
<i>umount</i> — mount and dismount filesystems	ConvexOS	MOUNT(8)
<i>uname</i> — get system information	ConvexOS	UNAME(2)
<i>uncompact</i> — compress and uncompress files, and cat them	ConvexOS	COMPACT(1)
<i>unexpand</i> — expand tabs to spaces, and vice versa	ConvexOS	EXPAND(1)
<i>ungetc</i> — push character back into input stream	ConvexOS	UNGETC(3S)
<i>uniq</i> — report repeated lines in a file	ConvexOS	UNIQ(1)
<i>units</i> — conversion program	ConvexOS	UNITS(1)
UNIX bootstrapping procedures	ConvexOS	REBOOT(8)
UNIX magtape interface	ConvexOS	MTIO(4)
<i>unlink</i> — remove a file or directory entry	ConvexOS	UNLINK(2)
unmap memory	ConvexOS	MUNMAP(2)
<i>unmount</i> — remove a file system	ConvexOS	UNMOUNT(2)
unmount a tape or deallocate a drive	ConvexOS	TPUNMOUNT(1)
<i>unsetenv</i> — manipulate environmental variables	ConvexOS	GETENV(3)
<i>update</i> — periodically update the super block	ConvexOS	UPDATE(8)
update date last modified of a file	ConvexOS	TOUCH(1)
update super-block	ConvexOS	SYNC(2)
update the super block	ConvexOS	SYNC(8)
<i>uptime</i> — show how long system has been up	ConvexOS	UPTIME(1)
user clock daemon	ConvexOS	CRON(1)
user database access	ConvexOS	GETPWUID(3)
user device (for debugging device drivers)	ConvexOS	UD(4)
user environment	ConvexOS	ENVIRON(7)
user information lookup program	ConvexOS	FINGER(1)
user interface to the notesfile system	ConvexOS	NFCOMMENT(3)
<i>USERFILE</i> — UUCP pathname permissions file	ConvexOS	USERFILE(5)
<i>users</i> — compact list of users who are on the system	ConvexOS	USERS(1)
<i>utime</i> — set file times	ConvexOS	UTIME(3C)
<i>utimes</i> — set file times	ConvexOS	UTIMES(2)
<i>utmp</i> — login records	ConvexOS	UTMP(5)
<i>uucico</i> — transfer files queued by uucp or uux	ConvexOS	UUCICO(8C)
<i>uuclean</i> — uucp spool directory clean-up	ConvexOS	UUCLEAN(8C)

Description	Product	Man Page
UUCP device description file	ConvexOS	L-DEVICES(5)
UUCP execution file interpreter	ConvexOS	UUXQT(8C)
UUCP hostname alias file	ConvexOS	L.ALIASES(5)
UUCP pathname permissions file	ConvexOS	USERFILE(5)
UUCP phone number index file	ConvexOS	L-DIAL.CODES(5)
UUCP remote command permissions file	ConvexOS	L.CMDS(5)
UUCP remote host description file	ConvexOS	L.SYS(5)
uucp spool directory clean-up	ConvexOS	UUCLEAN(8C)
<i>uucpd</i> — transfer files queued by uucp or uux	ConvexOS	UUCICO(8C)
<i>uuencode</i> — format of an encoded uuencode file	ConvexOS	UUENCODE(5)
<i>uulook</i> — monitor inter-system communications	ConvexOS	UULOOK(8)
<i>uupoll</i> — poll a remote UUCP site	ConvexOS	UUPOLL(8C)
<i>uusnap</i> — show snapshot of the UUCP system	ConvexOS	UUSNAP(8C)
<i>uuxqt</i> — UUCP execution file interpreter	ConvexOS	UUXQT(8C)
<i>va_arg</i> — variable argument list	ConvexOS	STDARG.H(3)
<i>va_end</i> — variable argument list	ConvexOS	STDARG.H(3)
<i>va_list</i> — variable argument list	ConvexOS	STDARG.H(3)
<i>va_start</i> — variable argument list	ConvexOS	STDARG.H(3)
<i>valloc</i> — aligned memory allocator	ConvexOS	VALLOC(3)
<i>varargs</i> — variable argument list	ConvexOS	VARARGS(3)
variable argument list	ConvexOS	STDARG.H(3)
variable argument list	ConvexOS	VARARGS(3)
<i>vdump</i> — incremental filesystem dump	ConvexOS	DUMP(8)
<i>verify</i> — verify characteristics of system files	ConvexOS	VERIFY(8)
<i>verify</i> — verify databases	ConvexOS	VERIFY(5)
verify characteristics of system files	ConvexOS	VERIFY(8)
verify databases	ConvexOS	VERIFY(5)
<i>vers</i> — set/display version numbers	ConvexOS	VERS(1)
<i>versatec</i> — versatec filters for use with lpr	ConvexOS	VERSATEC(8)
versatec filters for use with lpr	ConvexOS	VERSATEC(8)
<i>vfont</i> — font formats for the Benson-Varian or Versatec	ConvexOS	VFONT(5)
<i>vfork</i> — spawn new process in a virtual memory efficient way	ConvexOS	VFORK(2)
<i>vsprintf</i> — stdarg style formatted output conversion	ConvexOS	VPRINTF(3S)
<i>vhangup</i> — virtually “hangup” the current control terminal	ConvexOS	VHANGUP(2)
<i>vi</i> — screen oriented (visual) text editors based on ex	ConvexOS	VI(1)
<i>view</i> — screen oriented (visual) text editors based on ex	ConvexOS	VI(1)
<i>vipw</i> — edit the password and password restrictions files	ConvexOS	VIPW(8)
virtually “hangup” the current control terminal	ConvexOS	VHANGUP(2)
<i>vlimit</i> — control maximum system resource consumption	ConvexOS	VLIMIT(3C)
VME Dual SMD/ESDI Mass storage disk interface	ConvexOS	DD(4)
<i>vmstat</i> — report virtual memory statistics	ConvexOS	VMSTAT(1)
<i>vprintf</i> — stdarg style formatted output conversion	ConvexOS	VPRINTF(3S)
<i>vsprintf</i> — stdarg style formatted output conversion	ConvexOS	VPRINTF(3S)
<i>vtimes</i> — get information about resource utilization	ConvexOS	VTIMES(3C)

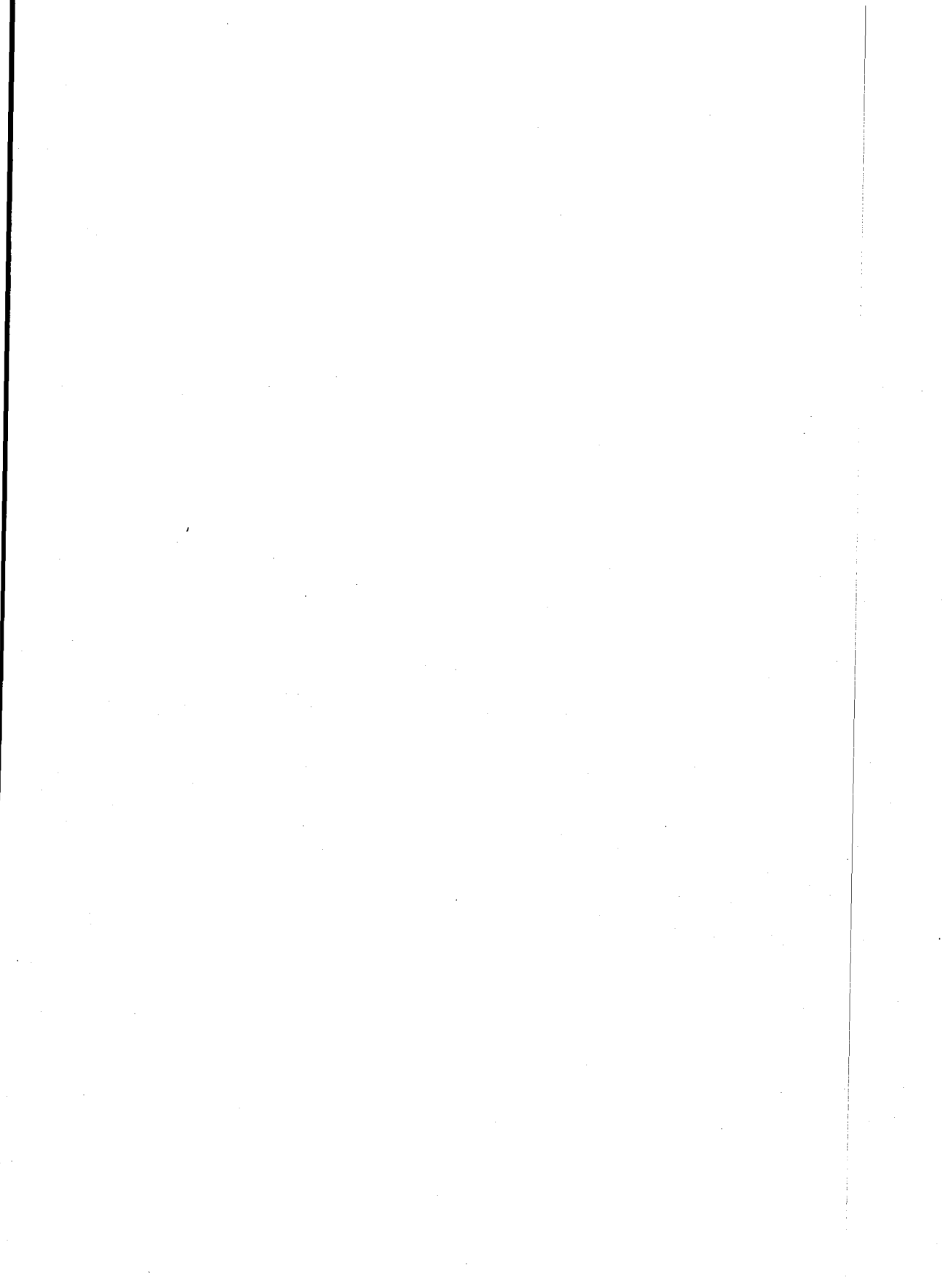
Description	Product	Man Page
<i>w</i> — who is on and what they are doing	ConvexOS	W(1)
<i>wait</i> — await completion of process	ConvexOS	WAIT(1)
<i>wait</i> — wait for process to terminate	ConvexOS	WAIT(2)
wait for a tpmount to complete	ConvexOS	TPWAIT(1)
wait for process to terminate	ConvexOS	WAIT(2)
wait for signal	ConvexOS	SIGSUSPEND(3)
wait, then return asynchronous I/O byte count	ConvexOS	ASIOSTAT(2)
<i>wait\$</i> — wait for process to terminate	ConvexOS	WAIT(2)
<i>waitpid</i> — wait for process to terminate	ConvexOS	WAIT(2)
<i>wall</i> — write to all users	ConvexOS	WALL(1)
<i>wc</i> — word count	ConvexOS	WC(1)
<i>wchar_t</i> — header file contains standard definitions	ConvexOS	STDDEF.H(3)
<i>wcstombs</i> — multibyte and wide character functions	ConvexOS	MBSTOWCS(3)
<i>wctomb</i> — multibyte and wide character functions	ConvexOS	MBSTOWCS(3)
<i>what</i> — show what versions of object modules were used to construct a file	ConvexOS	WHAT(1)
<i>whatis</i> — describe what a command is	ConvexOS	WHATIS(1)
<i>whatis</i> — display on-line reference manual information	ConvexOS	MAN(1)
<i>which</i> — locate a program file including aliases and paths (csh only)	ConvexOS	WHICH(1)
<i>who</i> — who is on the system	ConvexOS	WHO(1)
who is my mail from?	ConvexOS	FROM(1)
who is on and what they are doing	ConvexOS	W(1)
who is on the system	ConvexOS	WHO(1)
<i>whoami</i> — print the effective or real current user ID	ConvexOS	WHOAMI(1)
<i>whois</i> — DARPA Internet user name directory service	ConvexOS	WHOIS(1C)
<i>window</i> — window environment	ConvexOS	WINDOW(1)
window environment	ConvexOS	WINDOW(1)
word count	ConvexOS	WC(1)
working directory name	ConvexOS	PWD(1)
<i>write</i> — write output on a file	ConvexOS	WRITE(2)
<i>write</i> — write to another user	ConvexOS	WRITE(1)
write a crash dump to tape	ConvexOS	CRASHDUMP(8)
write output on a file	ConvexOS	WRITE(2)
write share data base file entry	ConvexOS	PUTSHARES(3)
write to all users	ConvexOS	WALL(1)
write to another user	ConvexOS	WRITE(1)
<i>writew</i> — gather output to file	ConvexOS	WRITEV(2)
<i>wtmp</i> — login records	ConvexOS	UTMP(5)
<i>xdump</i> — fast incremental filesystem dump	ConvexOS	XDUMP(8)
<i>xstr</i> — extract strings from C programs to implement shared strings	ConvexOS	XSTR(1)
<i>xtab</i> — directories to export to NFS clients	ConvexOS	EXPORTS(5)
<i>y0</i> — bessel functions	ConvexOS	J0(3M)
<i>y1</i> — bessel functions	ConvexOS	J0(3M)
<i>yacc</i> — yet another compiler-compiler	ConvexOS	YACC(1)
<i>yes</i> — be repetitively affirmative	ConvexOS	YES(1)
<i>yesterday</i> — show yesterday's date	ConvexOS	YESTERDAY(1)
yet another compiler-compiler	ConvexOS	YACC(1)

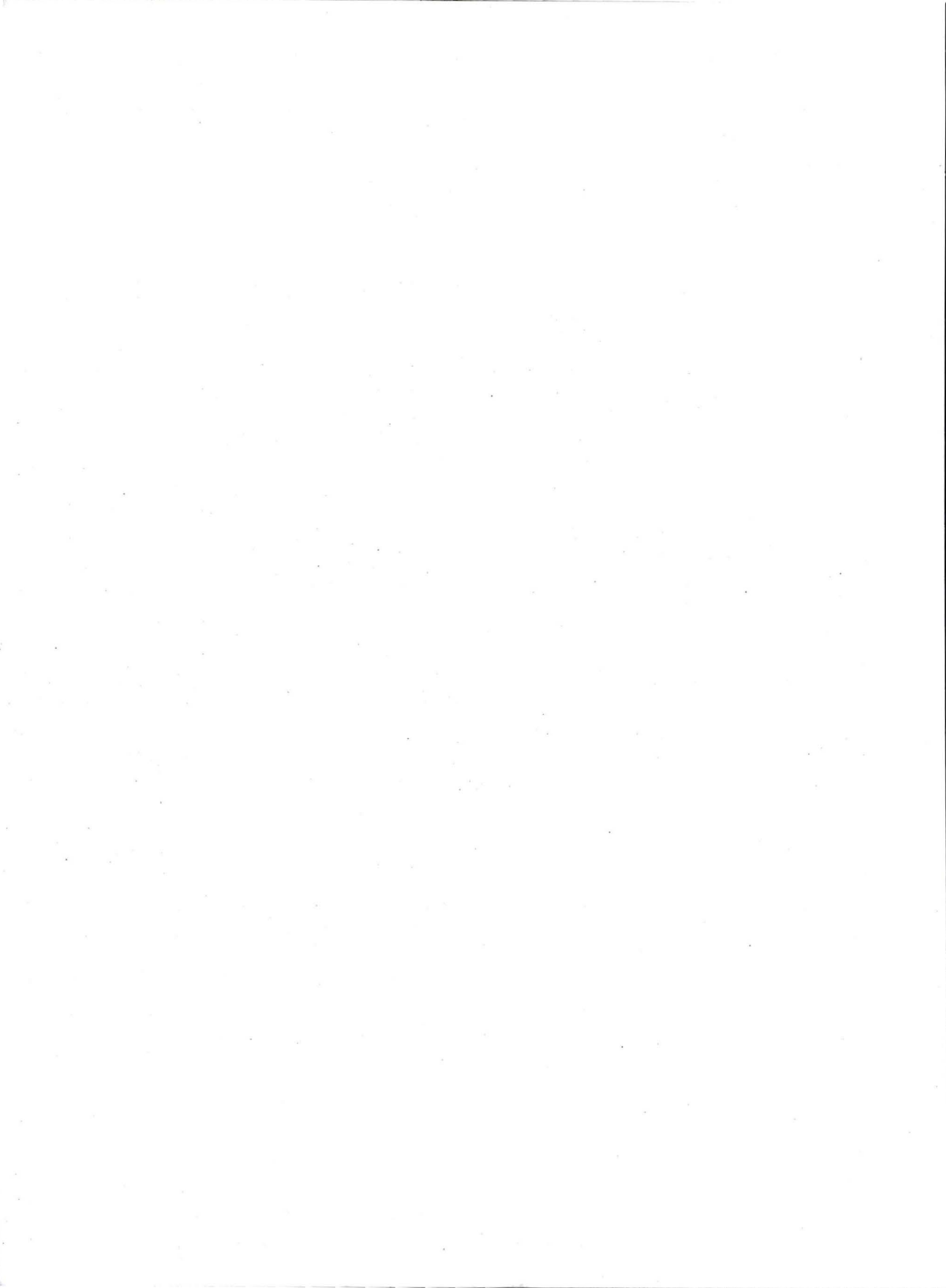
**Description****Product****Man Page**

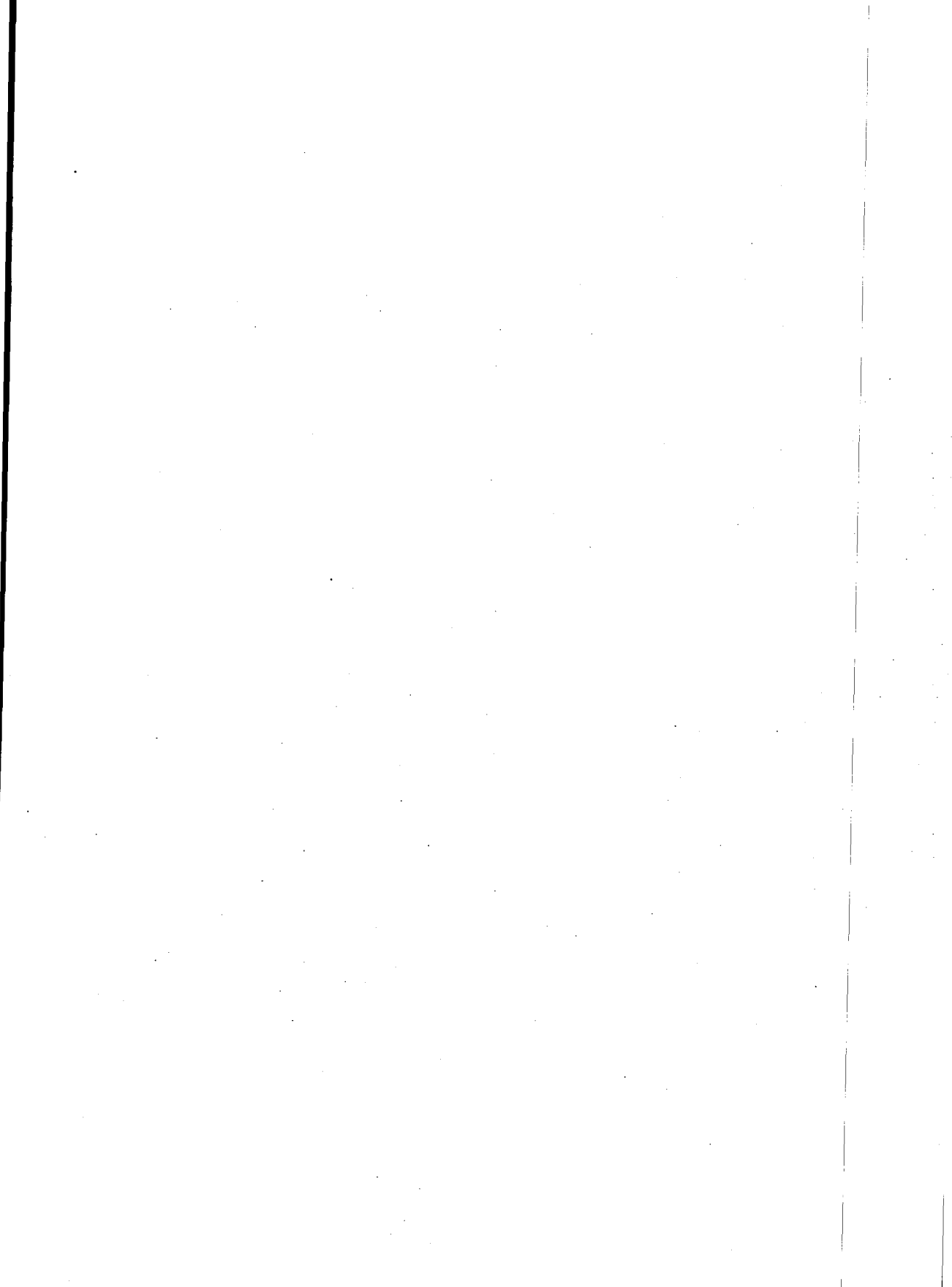
*yn* — *bessel functions*

ConvexOS

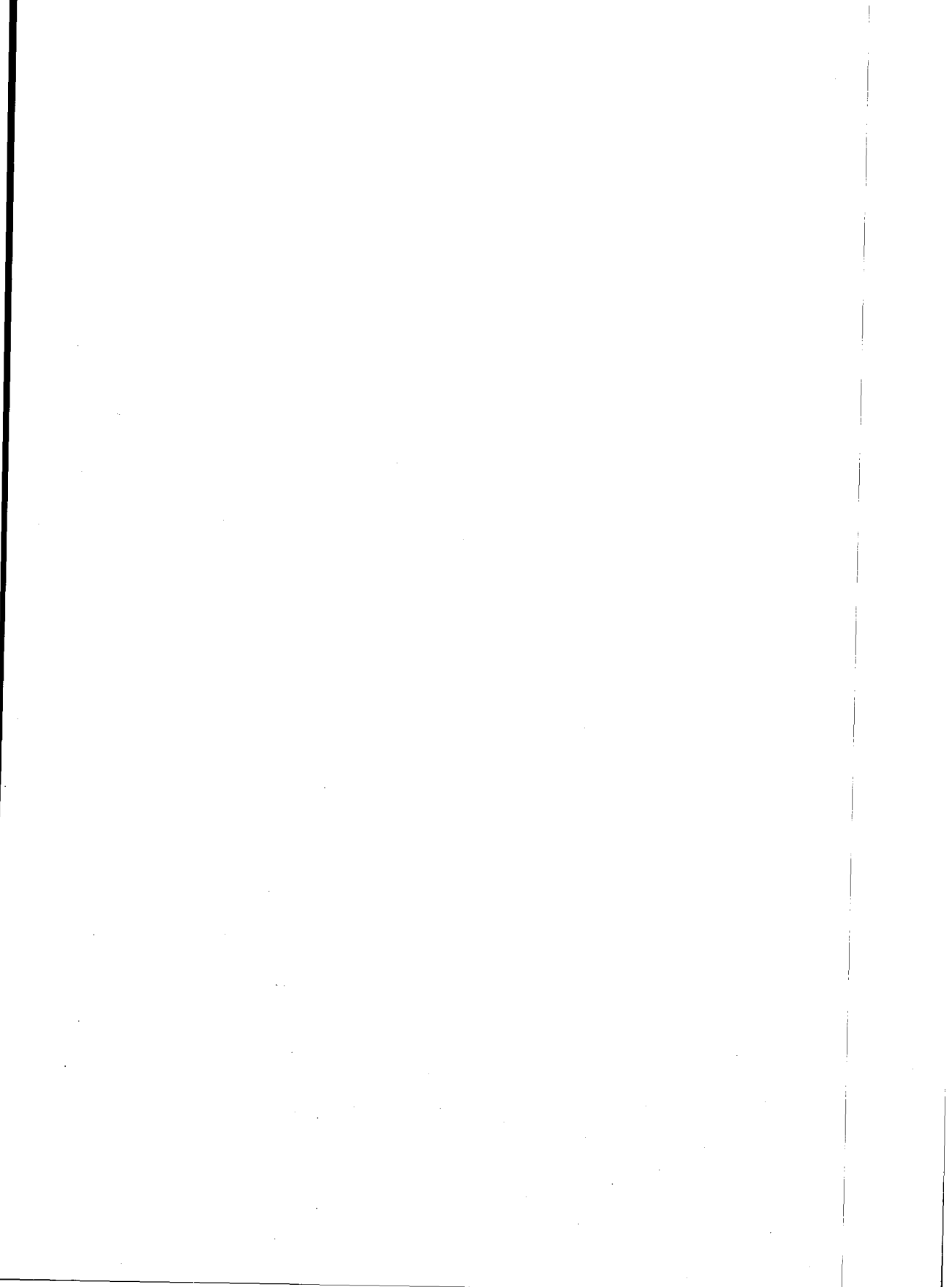
J0(3M)

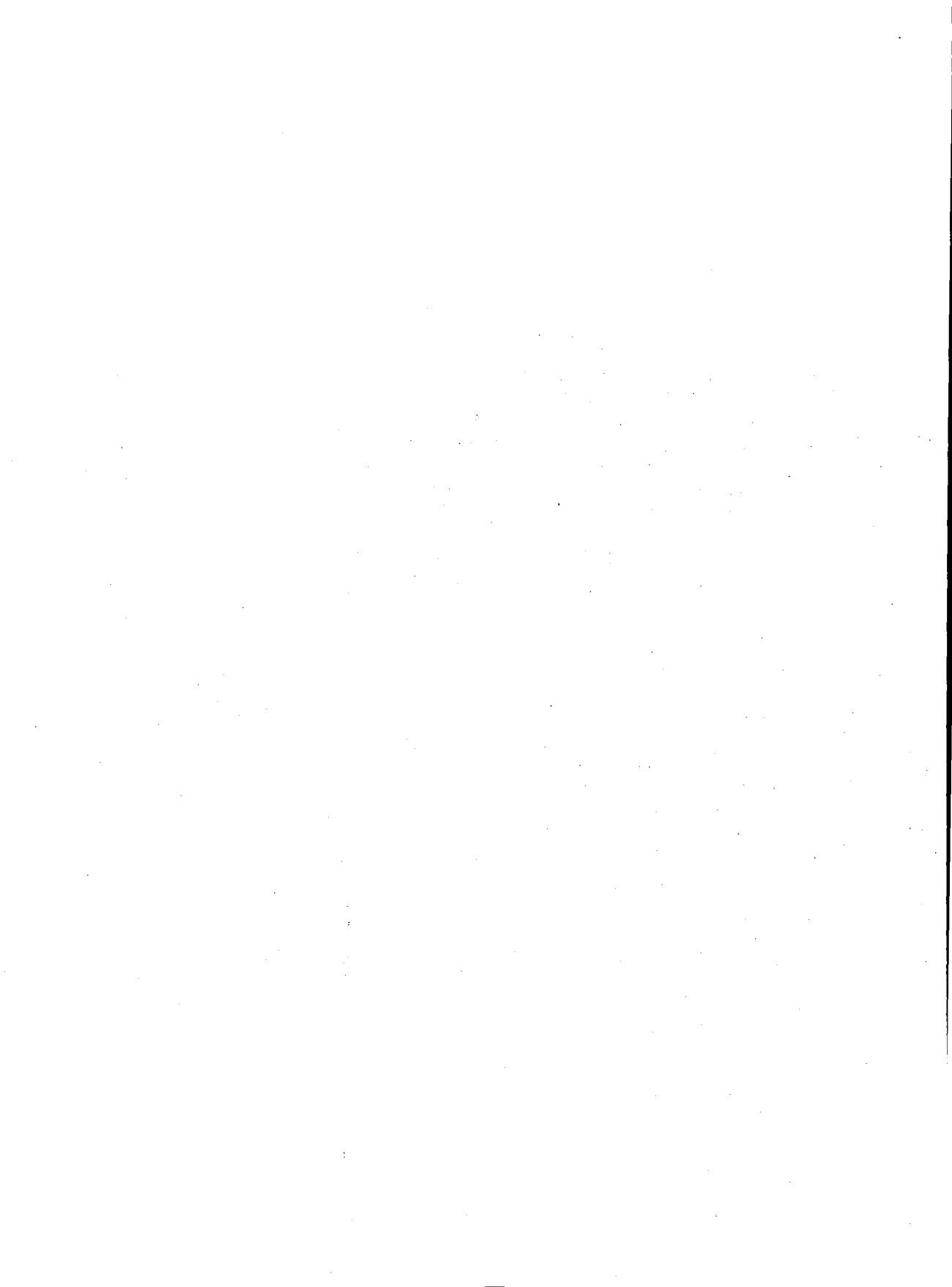


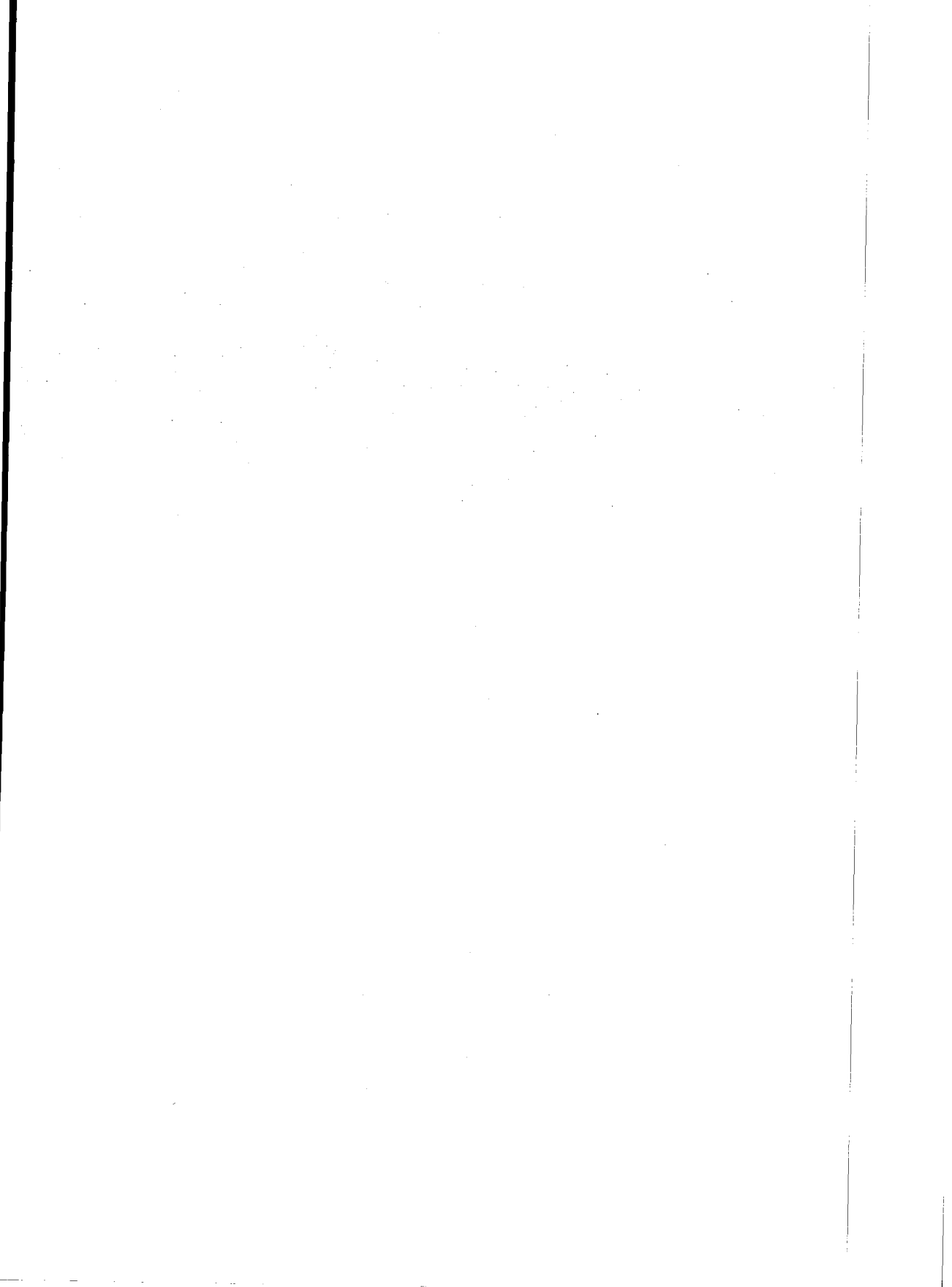


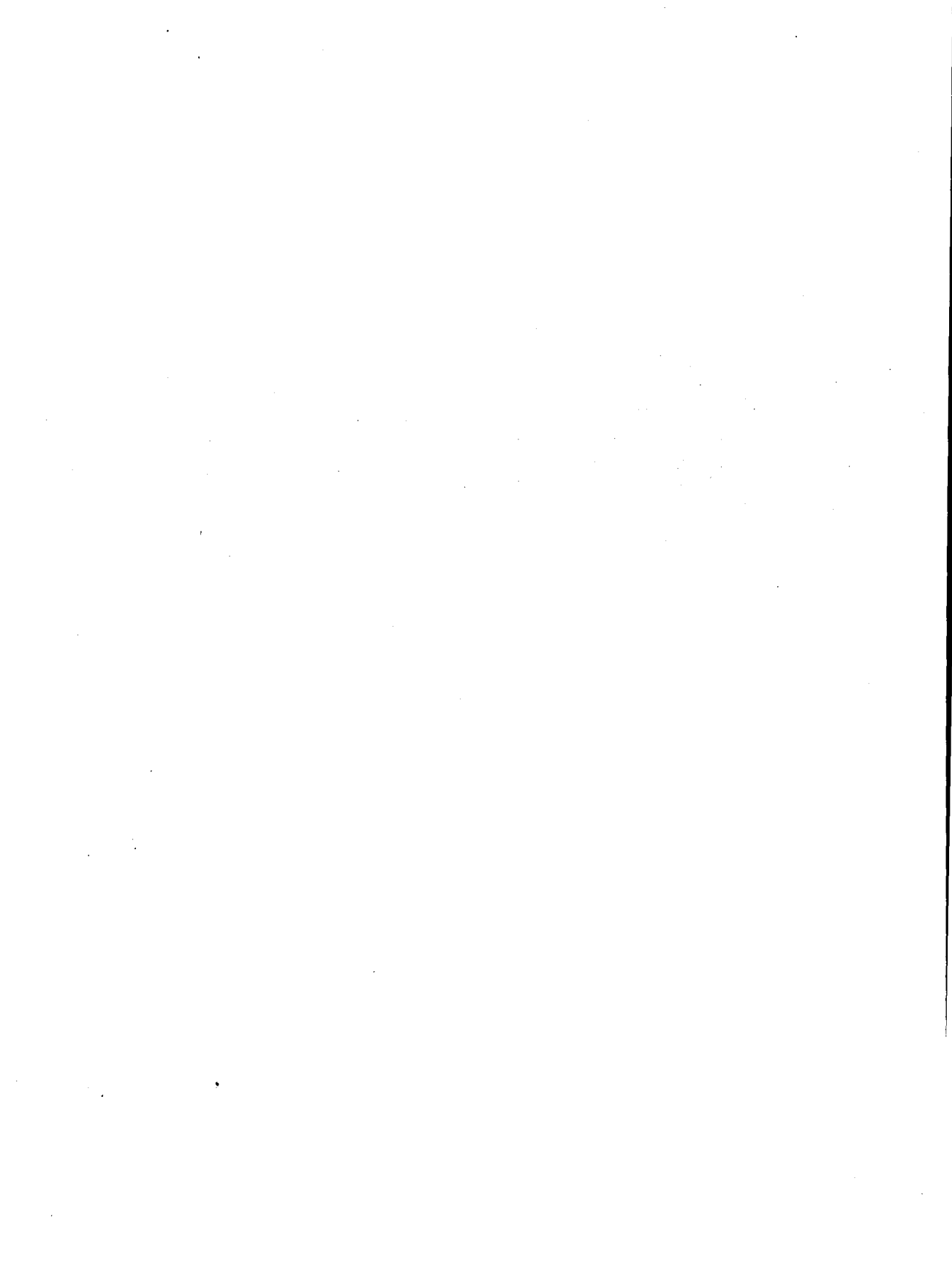


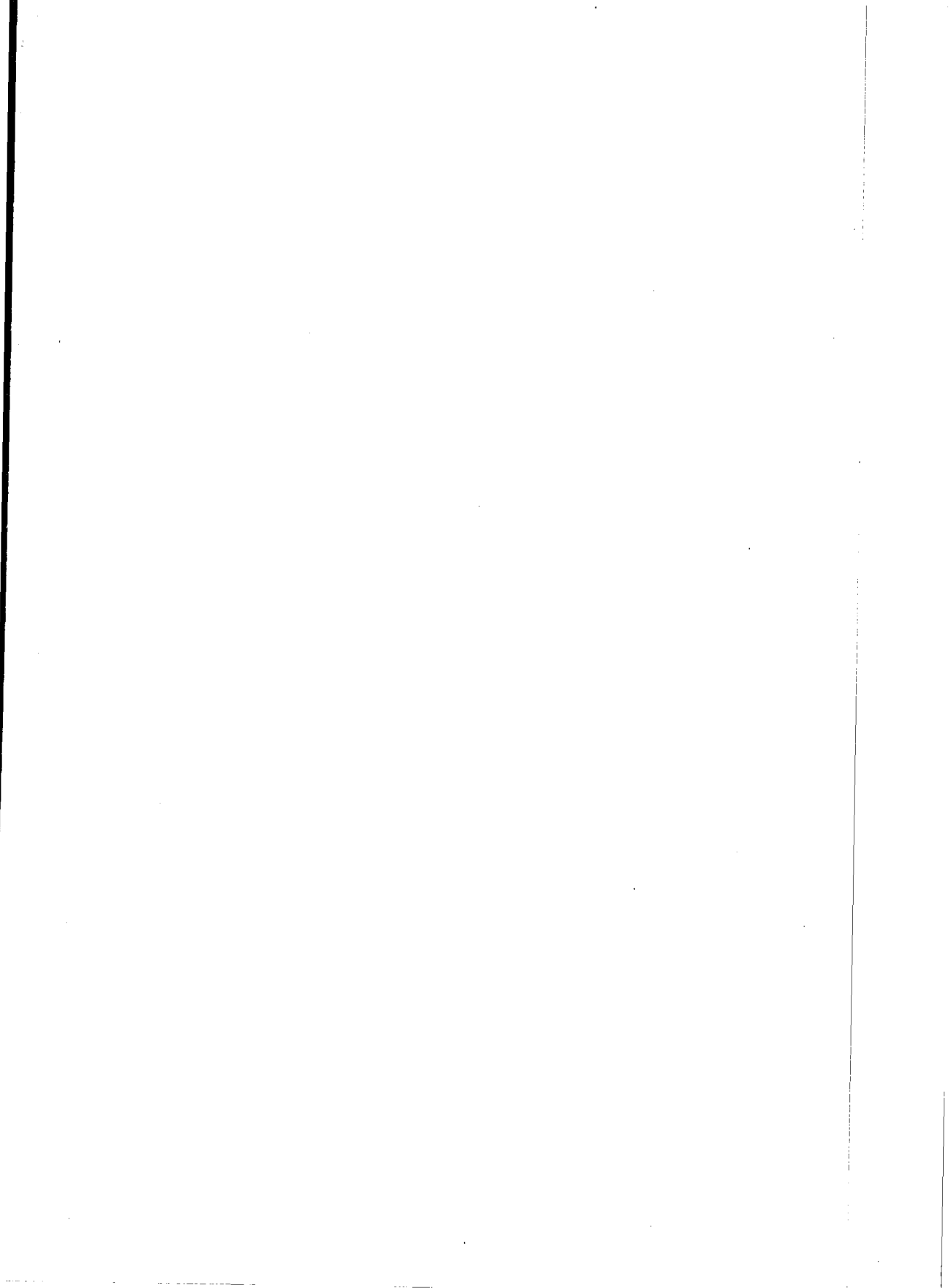




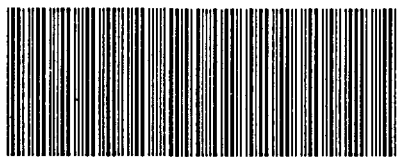








**Order Number**  
**DSW-333**



**Document Number**  
**710-015930-000**