CONCENTRIX-2800

Commands and Applications Manual

January 1992

The *Commands and Applications Manual* describes the utility commands and application programs available to non-privileged users of the Concentrix-2800 operating system.

Part number: 301-00505
Revision A

Model Number: D002.03

Use the model number for ordering purposes. The model number includes the text of the document and any ancillary parts such as binder, front and rear covers, and label. The part number refers to just this part - the text of the document.

Concentrix-2800 Version 2.2.0

ALLIANT COMPUTER SYSTEMS CORPORATION
One Monarch Drive ▪ Littleton ▪ Massachusetts 01460
508-486-4950

Alliant Computer Systems Corporation reserves the right to make changes in specifications and other information contained in this document without prior notice.

Although due care has been taken to present accurate information, ALLIANT DISCLAIMS ALL WARRANTIES WITH RESPECT TO THE CONTENTS OF THIS DOCUMENT (INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE), EITHER EXPRESSED OR IMPLIED. ALLIANT SHALL NOT BE LIABLE FOR DAMAGES RESULTING FROM ANY ERROR CONTAINED HEREIN, INCLUDING, BUT NOT LIMITED TO, FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF, OR IN CONNECTION WITH, THE USE OF THIS DOCUMENT.

WARNINGS: The software described in this document runs on the Alliant FX/2800 and FX/800 Series. This equipment generates, uses, and can radiate radio frequency energy, and, if not installed and used in accordance with instructions in the manual, may cause interference with radio communications.

It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, and with the Radio Interference Regulations, CRC c. 1374. These rules are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference, in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Le present appareil numerique n'emet pas de bruits radioelectriques depassant les limites applicables aux appareils numeriques de classes A prescrites dans le reglement sur le brouillage radioelectrique edicte par le ministere des communications du Canada.

The information contained in this document is summary in nature. More detailed information is available from Alliant.

This software product and its documentation set are copyrighted (unpublished) and all rights are reserved by Alliant Computer Systems Corporation. Usage of this product is only allowed under the terms set forth in the Alliant Purchase and License Agreement. Any reproduction or distribution of this document, in whole or in part, without the prior written consent of Alliant Computer Systems Corporation is prohibited.

Concentrix and Concentrix-2800 software and documentation is based in part on the Fourth Berkeley Software Distsribution under license from the The Regents of the University of California.

GNU Emacs software and documentation is based on the *GNU Emacs Manual* by Richard Stallman, 6th edition, version 18, March, 1987, distributed by the Free Software Foundation.

Copyright © 1992
ALLIANT COMPUTER SYSTEMS CORPORATION
All Rights Reserved

Revision History

	Date	OS Release	Comments
A	January 1992	2.2.0	new book for FX/2800 Series only

Trademarks of Alliant Computer Systems Corporation:

Alliant	FX/1	FX/Series	FX/FORTRAN
Concentrix	FX/4	FX/40	FX/C
Diagnostix	FX/8	FX/80	FX/Ada
DNX	VIP	FX/80T	Visualization Series
ANSR	ANSR/LCX	FX/82	FX/RT
VFX/4	VFX/40	VFX/80	VFX/82
Visual Supercomputer	DNX-2800	SRM/1	Visual Supercomputing
FX/2800 Series	FX/2800	Concentrix-2800	Scalar/C
FX/C-2800	FX/FORTRAN-2800	FX/RT-2800	Diagnostix-2800
FX/Ada-2800	FASTf	FX/800	CAMPUS
	CAMPUS/800	Clusternode	
Raster Technologies	GX4000	RNeWS	
Model One	Model One/10	Model One/25	Model One/40
Model One/60	Model One/75	Model One/85	Model One/380
ONELIB	ONELIBC	ONELIBF77	Model One/385

Other trademarks that may appear in this document include:

CCA EMACS	Trademark of Uniworks, Inc.
NeWS, NFS,NIS Sun-3, Sun-4, Sun-3/260 Sun-4/260	Trademarks of Sun Microsystems, Inc.
UNIX	Registered trademark of AT&T in the United States and Other Countries
Multibus, i860	Trademarks of Intel Corporation
DEC,DECnet, VAX, Digital Command Language (DCL), VAX/VMS, DNA, VAXcluster	Trademarks of Digital Equipment Corporation
Cray-1, Cray X-MP, Cray-2	Trademark of Cray Research, Inc.
Ethernet	Trademark of Xerox Corporation
Ada	Registered trademark of the U.S. government, Ada Joint Program Office (AJPO)
Network Computing System, NCS	Registered trademarks of Apollo Computer, Inc.
Hyperchannel	Trademark of Network Systems Corp.
X11, X-Window System	Trademark of Massachusetts Institute of Technology
Postscript	Registered trademark of Adobe Systems
Para-Linpack/FX, Para-Eispack/FX	Trademarks of Kuck and Associates, Inc.
CommUnity	Registered trademark of Technology Concepts, Inc.
UltraNet	Trademark of Ultra Network Technologies Corp.
VCL, EDT+	Boston Business Computing, Ltd.
KiNET, DEKnet	Trademarks of Ki Research, Inc.
EXPRESS	Trademark of ParaSoft Corporation

Printed in the United States of America

TABLE OF CONTENTS

Preface

Manual Objectives xiii
Audience xiii
Structure Of The Manual xiv
Characteristics Of Concentrix-2800 Commands xv
Changes In This Version Of The Manual xv
Related Documentation xvi
Documentation Conventions xvii

adb	general purpose debugger	1
addbib	create or extend bibliographic database (unsupported)	2
admin	create and administer SCCS files	4
afio	manipulate archives and files	8
alias	bind name to string (C shell)	11
alloc	show amount of dynamic core in use (C shell)	12
apply	apply a command to a set of arguments	13
apropos	find pertinent on-line documentation	14
ar	archive and library maintainer	15
as	Alliant FX/2800 Series assembler	17
at	execute commands at a later time	19
atq	print the queue of jobs waiting to be run	21
atrm	remove jobs queued by at	22
at_sign	set local numeric variable (C shell)	23
autopsy	utility to examine crash dumps	25
awk	pattern scanning and processing language	26
basename	strip path name prefix and suffix	29
bc	arbitrary-precision arithmetic language	30
bfio	manipulate archives and files	33
bg	execute a job in the background (C shell)	36
biff	be notified if mail arrives	37
binmail	send or receive mail among users	38
break	resume execution after foreach or while (C shell)	40
break	exit from enclosing for or while loop (Bourne shell)	41
breaksw	break from switch command (C shell)	42
cal	display calendar	43
calendar	reminder service	44
case	denote label in switch command (C shell)	45
case	control multi-way branching in shell scripts (Bourne shell)	46
cat	display text file	48
cb	C program beautifier	49

cc	C language compiler	50
ccat	display compressed files (old command, found in usr/old, replaced by zcat)	51
cd	change working directory(C shell)	52
cd	change working directory(Bourne shell)	53
cdc	change the delta commentary of an SCCS delta	54
chdir	change working directory	56
checkeq	error checker for mathematics typesetting (unsupported)	57
checknr	error checker for nroff/troff (unsupported)	58
chfn	change finger entry	60
chgrp	change file group	61
chkey	change your encryption key	62
chmod	change file access mode	63
chsh	change login shell	65
clear	clear terminal screen	66
cmp	compare two files	67
col	filter reverse line feeds	68
colcrt	filter nroff output for crt previewing (unsupported)	69
colrm	remove columns from a file	70
comb	combine SCCS deltas	71
comm	compare files	73
compact	compress files (old command, found in /usr/old, replaced by compress)	74
compress	compress files	75
continue	continue execution (C shell)	78
continue	continue execution (Bourne shell)	79
cp	copy files, directories, and subtrees	80
crypt	encode/decode files	81
csh	invoke C shell	83
ctags	create tags file	94
ctime	outputs the time in hexadecimal	96
date	display and set the date	97
dbx	source-level debugger	99
dc	desk calculator	101
dd	convert and copy a file	104
default	default case in switch command (C shell)	107
delta	make a delta (change) to an SCCS file	108
deroff	remove nroff, troff, tbl, eqn constructs (unsupported)	111
df	report free disk space on file systems	112
diction	find wordy sentences in document (unsupported)	114
diff	compare files and directories	115
diff3	three-way differential file comparison	118
dired	directory editor	120
dirs	display directory stack (C shell)	123
domainname	set or display name of current domain system	124
du	summarize disk usage	125
echo	echo arguments	126

ed	text editor	127
edit	simple text editor	128
egrep	search file for regular expression	130
else	alternate branch in if-then-else (C shell)	132
emacs	screen oriented display editor	133
end	terminate control structure (C shell)	140
endif	terminate if-then-else (C shell)	141
endsw	terminate switch (C shell)	142
enroll	enter password for encrypted mail	143
eqn	mathematics preprocessor for troff (unsupported)	144
error	analyze/disperse compiler error messages	145
eval	evaluate arguments and execute resulting comand	148
ex	text editor	149
exec	execute command in place of current shell (C shell)	150
exec	execute command in place of current shell (Bourne shell)	151
execute	run program on specific processor type	152
exit	exit from shell (C shell)	155
exit	exit from shell (Bourne shell)	156
expand	expand tabs to spaces	157
explain	interactive thesaurus for diction	158
expr	evaluate an expression	159
export	mark names for export (Bourne shell)	161
f	user information lookup program	162
false	return exit status value non-zero	163
fg	execute job in foreground (C shell)	164
fgrep	search file for pattern (fast)	165
file	determine file type	166
find	search directory tree for specific files	167
finger	user information lookup program	170
fmt	simple text formatter (unsupported)	172
fold	fold long lines for narrow output device	173
for	iterated loop construct (Bourne shell)	174
foreach	iterated loop construct (C shell)	176
fpr	convert Fortran carriage-control	178
from	display mail header lines	179
fsplit	split up multi-routine Fortran file	180
ftp	file transfer program	181
get	get a version of an SCCS file	190
glob	echo arguments delimited by nulls (C shell)	197
goto	unconditional branch control structure (C shell)	198
gprof	display call graph profile data	199
graph	draw a graph	202
grep	search a file for a pattern	204
groups	show group memberships	206
hashstat	print hash table statistics (C shell)	207
head	display first few lines of a file	208
help	describe on-line documentation	209

helpsccs	get help on SCCS commands and errors	211
history	display most recent commands (C shell)	212
hostid	set/display host system identifier	213
hostname	set/display host system name	214
if	simple two-way branch control structure (C shell)	215
if–then–else	multiple two-way branch control structure (C shell)	216
if	multiple two-way branch control structure (Bourne shell)	218
indent	indent and format C program source	220
indxbib	build inverted index for bibliography (unsupported)	225
info	documentation reader	226
install	install binaries	227
jobs	list active jobs (C shell)	228
join	relational database operator	229
kconfig	system configuration and memory usage	230
keylogin	decrypt and store secret key	232
kill	terminate a process (Bourne shell)	233
kill	terminate job or process (C shell)	234
last	indicate last logins of users and teletypes	235
lastcomm	show commands in reverse order	236
ld	link editor	237
leave	remind you when you have to leave	240
lex	generator of lexical analysis programs	241
limit	limit resource consumption (C shell)	242
lint	C program verifier	244
ln	make links	246
lock	reserve terminal	247
logger	make entries in the system log	248
login	create new user process	249
login	execute login arguments (Bourne shell)	251
logout	terminate login shell (C shell)	252
look	find lines in a sorted list	253
lookbib	find references in a bibliography (unsupported)	254
lorder	find ordering relation for object library	255
lpq	spool queue examination program	256
lpr	print files on line printer	258
lprm	remove jobs from line printer queue	261
lprof	line level profiler	263
lptest	generate line printer ripple pattern	265
ls	list contents of directory	266
m4	macro processor	269
machid	get machine id	271
mail	send and receive mail	272
make GNU_make	maintain program groups	277
make	maintain program groups (old)	282
man	read on-line documentation	285
mesg	permit or deny messages	289

mkdir	create directory	290
mkstr	create error message file from C source	291
mon	monitor process/system status (crt display)	293
more	file perusal filter for crt viewing	298
msgs	read system messages	301
mt	magnetic tape manipulating program	303
mv	move or rename files	305
netstat	show network status	306
neqn	mathematics preprocessor for nroff (unsupported)	309
nice	set priority for command (Bourne shell)	310
nice	set priority for command (C shell)	311
nm	dump object file symbol table	312
nohup	ignore hangup signals (Bourne shell)	313
nohup	ignore hangup signals (C shell)	314
notify	display job status changes (C shell)	315
nroff	text formatter (unsupported)	316
od	octal, decimal, hex, ASCII dump	318
on	execute a command remotely	321
onintr	enable interrupt handling (C shell)	322
page	file perusal filter for crt viewing	323
pagesize	display system page size	326
passwd	change login password, finger information, or default shell	327
percent_sign	place job in foreground/background (C shell)	329
plot	graphics filters	330
popd	pop directory stack (C shell)	333
pr	format and display file	334
prelf	dump contents of an ELF file as text	335
print	pr to the line printer	336
printenv	display environment variables	337
prs	print an SCCS file	338
ps	display process status information	342
ptx	permuted index (unsupported)	345
pushd	push directory onto stack (C shell)	347
pwd	display working directory pathname	348
qdel	delete batch request	349
qstat	show batch request status	350
qsub	submit batch request	351
quota	display disk usage and limits	354
ranlib	convert archives to random-access libraries	355
ratfor	"rational Fortran" preprocessor	356
rcp	remote file copy	358
rdist	remote file distribution program	360
read	read one line from input and assign words to variable (Bourne shell)	365
readonly	given names are marked readonly (Bourne shell)	366
refer	find/insert literature references (unsupported)	367
rehash	recompute path table (C shell)	370

repeat	execute command repetitively (C shell)	371
reset	reset the teletype bits to a sensible state	372
reten	retension tape cartridge	373
rev	reverse lines of a file	374
rlogin	remote login	375
rm	remove (unlink) files	377
rmail	handle remote mail received via uucp	378
rmdel	remove a delta from an SCCS file	379
rmdir	remove (unlink) directories	380
rpcgen	an RPC protocol compiler	381
roffbib	display bibliographic database (unsupported)	384
rsh	remote shell	385
rt	read magnetic tape and print record sizes	387
rup	show host status of local machines (RPC version)	388
ruptime	show host status of local machines	389
rusers	who's logged in on local machines (RPC version)	390
rwall	write to all users over a network	391
rwho	show who's logged in on local machines	392
scc	Scalar/C compiler command	393
sccs	Source Code Control System (Berkeley interface)	397
sccsdiff	compare two versions of an SCCS file	401
script	make typescript of terminal session	402
sed	stream editor	403
sendbug	bug reporting utility	406
set	set local variable (C shell)	408
set	set shell options (Bourne shell)	411
setenv	set environment variable (C shell)	412
sh	invoke Bourne shell	414
shift	shift members of variable (C shell)	422
shift	rename positional parameters (Bourne shell)	423
size	display size of an object file	424
sleep	suspend execution for an interval	425
soelim	eliminate .so's from nroff input	426
sort	sort or merge files	427
sortbib	sort bibliographic database (unsupported)	429
source	execute commands from file (C shell)	430
spell	find spelling errors	431
spellin	create new hash list for spell	433
spellout	report on hash list for spell	434
spline	interpolate smooth curve	435
split	split a file into pieces	436
stop	stop background job (C shell)	437
strings	find printable strings in a binary file	438
strip	remove symbols and relocation bits	439
stty	set terminal options	440
style	analyze document surface characteristics (unsupported)	444
su	substitute userid (superuser)	445

sum	sum and count blocks in a file	446
suspend	stop current shell (C shell)	447
switch	multi-way branch control structure (C shell)	448
symorder	rearrange name list	450
syscfg	show system configuration	451
tabs	set terminal tabs	452
tail	display the last part of a file	453
talk	talk to another user	454
tar	tape archiver	456
tbl	format tables for nroff or troff (unsupported)	459
tc	phototypesetter simulator	460
tcopy	make a copy of a magnetic tape, or print information about tape format	461
tcpdump	dump traffic on a network	462
tee	copy to standard output and to file	475
telnet	user interface to the TELNET protocol	476
test	evaluate expression and return status	481
tftp	trivial file transfer program	483
time	time a command (C shell)	485
time	time a command	487
times	print accumulated user and system times (Bourne Shell)	488
tip	connect to a remote system	489
touch	update date last modified of a file	493
tr	translate characters	494
trap	read and execute command upon receiving signal (Bourne Shell)	495
troff	text formatting and typesetting (unsupported)	496
true	provide truth values	498
tset	terminal initialization	499
tsort	topological sort	504
tty	get terminal name	505
ul	convert underlining for terminal	506
umask	set file creation mask (C shell)	507
umask	set file creation mask (Bourne shell)	508
unalias	remove alias (C shell)	509
uncompact	expand files that were compressed with the compact utility (old command, found in /usr/old, replaced by uncompress)	510
uncompress	expand files that were compressed with the compress utility	511
unexpand	compress spaces to tabs	513
unhash	disable hash table (C shell)	514
unifdef	remove ifdef lines from a C program	515
uniq	report repeated lines in a file	517
units	quantitative conversion program	518
unlimit	remove resource limitation (C shell)	519
unset	remove local variable (C shell)	520
unsetenv	remove environment variable (C shell)	521
uptime	show how long system has been up	522
users	display list of current system users	523

uucp	unix to unix copy	524
uudecode	decode binary file from mail	527
uuencode	encode binary file for mail	528
uulog	display UUCP log files	529
uuname	list names of UUCP hosts	530
uuq	examine or manipulate the uucp queue	531
uusend	send file to remote host	532
uux	unix to unix command execution	533
vacation	return an "I am on vacation" indication	536
val	validate SCCS file	538
vi	screen oriented (visual) display editor	540
vmstat	report virtual memory statistics	542
w	show who is on and what they are doing	543
wait	wait for background jobs (C shell)	545
wait	await completion of process (Bourne shell)	546
wall	write to all users	547
wc	count words, lines, and characters	548
what	identify SCCS files	549
whatis	show description of word	550
whatisin	show contents of on-line documentation section	551
whereis	locate source, binary, and on-line documentation files	553
which	show pathname for command (C shell)	555
while	test–and–loop control structure (C shell)	556
while	test–and–loop control structure (Bourne shell)	558
who	show who is on the system	559
whoami	display effective current user id	560
whois	DARPA Internet user name directory service	561
window	a windowing environment	562
write	write to another user	571
xget	read encrypted mail	572
xproc	process time and cpu monitor	573
xsend	send encrypted mail	575
xstr	extract strings from C programs	576
yacc	yet another compiler-compiler	578
yes	be repetitively affirmative	579
ypcat	display values in an NIS data base	580
ypmatch	print the value of one or more keys from a yp map	582
yppasswd	change login password in network information service	583
ypwhich	which host is the YP server or map master?	584
zcat	expand compressed files to standard output	585

PREFACE

This manual describes all of the commands and application programs available to non-privileged users of the Concentrix-2800 operating system, including commands that are built into the C shell and Bourne shell. It is intended for all system users. It corresponds to Section 1 of the *UNIX Programmer's Guide*.

MANUAL OBJECTIVES

The objective of this manual is to provide a brief, utilititarian description of each operating system command and application program. Some programs are documented in full elsewhere; for others this manual is the only source of information. Thus, some programs are described more thoroughly than others.

AUDIENCE

This manual is intended for all system users. No prior knowledge is required.

STRUCTURE OF THE MANUAL

This manual consists of entries or "manual pages" in alphabetic order. Each entry is based on the following format. Empty subsections are omitted.

- **name**

  The name of the command is shown in large, bold type.

- **brief description**

  A brief description of its purpose is shown directly below the name. This description is also used as the database for the program whatis.

- **SYNTAX**

  The syntax diagram summarizes the use of the command.

- **ARGUMENTS**

  This subsection describes the meanings of the arguments shown in the syntax diagram.

- **DESCRIPTION**

  This subsection discusses the command in more detail.

- **NOTES**

  This subsection describes known problems, restrictions, and miscellaneous information.

- **EXAMPLE**

  This subsection shows one or more examples, which illustrate how the command is used.

- **DIAGNOSTICS**

  This subsection discusses the diagnostic messages that may be produced by the command. Messages that are intended to be self-explanatory are not listed.

- **FILES**

  This subsection gives the names of files that the command references or depends upon.

- **SEE ALSO**

  This subsection gives cross-references to related information in this and other documents. Be sure to check here if you find yourself bewildered.

CHARACTERISTICS OF CONCENTRIX-2800 COMMANDS

Commands generally reside in directory /bin (for *bin*ary programs). Some programs also reside in /usr/bin, or in /usr/ucb, to save space in /bin. Command directories are searched automatically by the command interpreters.

Most commands do their work silently. There is no prompting or chatter, and error messages are occasionally curt. This terseness is sometimes disconcerting to new users, but many experienced users find it desirable.

Although most commands conform to a general syntactic model, there are subtle variations:

- Some commands allow more than one option to follow a single delimiter; others do not. For example, these are not necessarily equivalent:

```
command -a -b -c
command -abc
```

- Some commands use the hyphen (-) to delimit options; others use the plus sign (+).

- The significance of "white space" (spaces and tabs) between options and arguments varies among commands. Some require it, some prohibit it, and some treat it as optional. For example, these are not necessarily equivalent:

```
command -f name
command -fname
```

There are a few commands that bear little resemblance to the general model. For example, the find command has a completely unique syntax.

Some options, particularly single characters, have entirely different meanings from one command to another. For example:

```
command1 -c
command2 -c
```

The -c option might stand for "column" in one command and "character" in the other.

CHANGES IN THIS VERSION OF THE MANUAL

Version GA of the *Concentrix Commands and Applications* manual described both Version 5.7 of the Concentrix operating system, and Version 2.1 of the Concentrix-2800 operating system. The vast majority of the functions found in the BSD4.3 (UNIX) operating system have been implemented.

Version A of *Concentrix-2800 Commands and Applications* describes only the Concentrix-2800 operating system, version 2.2.

The information that has been added, changed, or deleted since the release of Concentrix-2800 2.1 is listed below. See the *Concentrix-2800 2.2 Installation Guide and Release Notes* for complete information on the changes and enhancements available in this version of the operating system.

Commands Added

binmail	send or receive mail among users (version 7 UNIX mail)
chkey	change your encryption key
keylogin	descrypt and store secret key
machid	get machine identification
ptx	permuted index
quota	display disk usage and limits
tcpdump	dump traffic on a network

Commands Changed

autopsy	options added
du	added new conv options, note
execute	change to -ce option
find	added options -fstype, -prune; info on updatedb
last	added option -f
ld	added -nc option and note on new functionality
mail	new version, added section on mail addressing
make	changed to GNU_make, which offers many more options
mon	new options
msgs	new option -c
rcp	changed information on hostnames, domains
rlogin	changed information on hostnames
rpcgen	extensive additions
rusers	added file reference to /etc/inetd.conf
syscfg	added information on new display
telnet	now supports 'negotiate window size' option
vacation	added options
which	added -c option
ypcat	added options
yppasswd	added ypchfn option for NIS

Commands Removed

prmail	old; not used
vgrind	not supported; does not work
vtroff	not supported; does not work

RELATED DOCUMENTATION

This section lists all of the documents that you receive with your system.

Name	Model Number
Alliant Documentation Map	D000.00
FX/2800 Series System Description	D000.01
FX/2800 Series Site Preparation Guide	D000.02*
FX/800 Site Preparation Guide	D000.03*
SRM/1 Site Preparation Guide	D000.04*
FX/2800 Series Porting Guide	D000.05
Intel i860 Programmer's Reference Guide	D000.07
FX/2800 Series Diagnostics User's Guide	D000.08
Concentrix-2800 Installation Guide and Release Notes	D002.00
Concentrix-2800 Commands and Applications Manual	D002.03
Concentrix System User's Guide	D002.05
Concentrix Document Preparation Guide	D002.06
GNU Emacs Manual	D002.07
Practical Guide to UNIX	D002.08
Learning the UNIX Operating System	D002.09
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files and File Formats	D002.10
Concentrix-2800 System Reference: System Calls and C Library Routines	D002.12
Concentrix-2800 Programming Tools Guide	D002.14
The C Programming Language	D002.15
Scalar/C Programmer's Handbook	D002.16
Concentrix-2800 Systems Programming Guide	D002.17
Concentrix-2800 System Administration	D002.18
FX/2800 Series Common Library	D002.21

* you receive the appropriate manual for your site.

You can order additional copies of these manuals. You can also order these manuals in various kits. See D000.00 – *Alliant Documentation Map*.

DOCUMENTATION CONVENTIONS

The "UNIX Programmer's Manual," on which this manual is based, was written by many people over many years. Although an effort has been made to use a consistent set of documentation conventions, you may notice variations.

Alliant uses the following documentation conventions for the different types of text noted below:

Command Syntax

boldface	indicates a constant (option); type it exactly as it appears.
italic	indicates a variable; replace it with an appropriate value.
[ ]	(brackets) indicate an optional item or set of items.
{ }	(braces) indicate a set of items from which you must select one.
...	indicates that the previous argument-prototype can be repeated.

Normal Text

sans serif	indicates a keyword; the name of a command, file name, or data item.
italic	indicates a variable.
CTRL *c*	represents a control character. You type it by holding down the control key and pressing the key for *c* at the same time. For example, to type CTRL D hold down the control key and press **D** at the same time.
RETURN	represents the return key, which is used to terminate all user input lines unless specifically indicated.

Examples

`monospace`	smaller type indicates output from the computer.
`monospace`	larger type indicates input by the user.

adb

general purpose debugger

Adb is a general purpose debugging program. It can be used to examine core dumps, to provide a controlled environment for the execution of programs, and to patch files. Requests to adb are read from the standard input and responses are written to the standard output.

---

adb [-w] [-I *dir*] [*objfil*] [*corfil*] [*libfil*]

ARGUMENTS

-w creates both *ob-file* and *core-file* if necessary and opens them for reading and writing.

-I *dir* specifies a directory where files to be read with $< or $<< are sought; the default is /usr/lib/adb.

objfil is normally an executable program file, preferably containing a symbol table; if not, the symbolic features of *adb* cannot be used although the file can still be examined. The default for *objfil* is a.out.

corfil is assumed to be a core image file produced after executing *objfil*; the efault for *corfil* is core. A hyphen (-) inhibits the loading of any core file.

libfil is a library file to be used during debugging. The default for *libfil* is /lib/shared.

To exit from adb, use CTRL D or the $q command. Adb ignores QUIT signals; INTERRUPT signals reset adb to command mode.

DESCRIPTION

See the *Concentrix-2800 Systems Programming Guide* for a full description of adb.

SEE ALSO

as, cc, dbx, scc
Concentrix-2800 Systems Programming Guide, adb
Concentrix-2800 System Reference Manual: System Calls and C Library Subroutines: ptrace

addbib

create or extend bibliographic database (unsupported)

The addbib command prompts you for various bibliographic fields, reads responses from the terminal, and sends output records to a database. When addbib starts up, answering y to the initial `Instructions?` prompt yields directions; typing n or RETURN skips them.

---

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

ARGUMENTS

-p *promptfile*
causes addbib to use a new prompting skeleton, defined in *promptfile*. This file should contain prompt strings, a tab, and the key-letters to be written in the *database*.

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

database specifies a bibliographic database.

DESCRIPTION

A null response (RETURN) to a field prompt means to leave out that field. A hyphen (-) means to go back to the previous field. A trailing backslash allows a field to be continued on the next line. The repeating `Continue?` prompt allows you either to resume by typing y or RETURN, to quit the current session by typing n or q, or to edit the database with any system editor (vi, ex, edit, ed).

The most common key-letters and their meanings are given below. Addbib insulates you from these key-letters, since it gives you prompts in English, but if you edit the bibliography file later on, you will need to know this information.

%A Author's name
%B Book containing article referenced
%C City (place of publication)
%D Date of publication
%E Editor of book containing article referenced
%F Footnote number or label (supplied by *refer*)
%G Government order number
%H Header commentary, printed before reference
%I Issuer (publisher)
%J Journal containing article
%K Keywords to use in locating reference
%L Label field used by -k option of *refer*
%M Bell Labs Memorandum (undefined)
%N Number within volume

%O Other commentary, printed at end of reference
%P Page number(s)
%Q Corporate or Foreign Author (unreversed)
%R Report, paper, or thesis (unpublished)
%S Series title
%T Title of article or book
%V Volume number
%X Abstract – used by *roffbib*, not by *refer*
%Y,Z ignored by *refer*

Except for A, each field should be given just once. Only relevant fields should be supplied.

EXAMPLE

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

FILES

promptfile optional file to define prompting

SEE ALSO

refer, sortbib, roffbib, indxbib, lookbib
Document Preparation Guide: Refer – A Bibliography System

admin

create and administer SCCS files

The admin command is used to create new SCCS files and change parameters of existing ones.

---

admin [*option...*] *files*

ARGUMENTS

Arguments to admin, which can appear in any order, consist of keyletter arguments beginning with a hyphen and named files beginning with the characters "s.". Each keyletter argument is described as though only one named file is to be processed since the effects of the arguments apply independently to each named file.

-n indicates that a new SCCS file is to be created.

-i[*name*] specifies the *name* of a file from which the text for a new SCCS file is to be taken. The text constitutes the first delta of the file (see -r keyletter for delta numbering scheme). If the i keyletter is used, but the file name is omitted, the text is obtained by reading the standard input until an end-of-file is encountered. If this keyletter is omitted, the SCCS file is created empty. Only one SCCS file can be created by an admin command on which the i keyletter is supplied. Using a single admin to create two or more SCCS files require that they be created empty (no -i keyletter). Note that the -i keyletter implies the -n keyletter.

-r*rel* specifies the *rel*ease into which the initial delta is inserted. This keyletter can be used only if the -i keyletter is also used. If the -r keyletter is not used, the initial delta is inserted into release 1. The level of the initial delta is always 1 (by default initial deltas are named 1.1).

-t[*name*] specifies the *name* of a file from which descriptive text for the SCCS file is to be taken. If the -t keyletter is used and admin is creating a new SCCS file (the -n and/or -i keyletters also used), the descriptive text file name must also be supplied. In the case of existing SCCS files: a -t keyletter without a file name causes removal of descriptive text (if any) currently in the SCCS file, and a -t keyletter with a file name causes text (if any) in the named file to replace the descriptive text (if any) currently in the SCCS file.

-f*flag*[*value*] specifies a *flag*, and, possibly, a value for the *flag*, to be placed in the SCCS file. Several f keyletters can be supplied on a single admin command line. The allowable *flag*s and their values are:

- **b** allows use of the -b keyletter on a *get* command to create branch deltas.
- **c***ceil* specifies the highest release (i.e., "ceiling"), a number less than or equal to 9999, that can be retrieved by a get command for editing. The default value for an unspecified c flag is 9999.

f*floor* specifies the lowest release (i.e., "floor"), a number greater than 0 but less than 9999, that can be retrieved by a get command for editing. The default value for an unspecified f flag is 1.

d*SID* specifies the default delta number (SID) to be used by a get command.

i causes the `No id keywords (ge6)` message issued by get or delta to be treated as a fatal error. In the absence of this flag, the message is only a warning. The message is issued if no SCCS identification keywords (see *get*) are found in the text retrieved or stored in the SCCS file.

j allows concurrent get commands for editing on the same SID of an SCCS file. This allows multiple concurrent updates to the same version of the SCCS file.

l*list* specifies a *list* of releases to which deltas can no longer be made (get –e against one of these "locked" releases fails). The *list* has the following syntax:

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

The character a in the *list* is equivalent to specifying *all releases* for the named SCCS file.

n causes delta to create a "null" delta in each of those releases (if any) being skipped when a delta is made in a *new* release (e.g., in making delta 5.1 after delta 2.7, releases 3 and 4 are skipped). These null deltas serve as "anchor points" so that branch deltas can later be created from them. The absence of this flag causes skipped releases to be non-existent in the SCCS file preventing branch deltas from being created from them in the future.

q*text* specifies user definable text substituted for all occurrences of the %Q% keyword in SCCS file text retrieved by get.

m*mod* specifies the *mod*ule name of the SCCS file substituted for all occurrences of the %M% keyword in SCCS file text retrieved by *get*. If the m flag is not specified, the value assigned is the name of the SCCS file with the leading s. removed.

t*type* specifies the *type* of module in the SCCS file substituted for all occurrences of %Y% keyword in SCCS file text retrieved by *get*.

v[*pgm*] causes delta to prompt for Modification Request (*MR*) numbers as the reason for creating a delta. The optional value specifies the name of an *MR* number validity checking program (see *delta*). (If this flag is set when creating an SCCS file, the m keyletter must also be used even if its value is null).

-d*flag*[*value*]

causes removal (deletion) of the specified *flag* from an SCCS file. The -d keyletter can be specified only when processing existing SCCS files. Several -d keyletters can be supplied on a single admin command. See the -f keyletter for allowable *flag* names.

l*list* A *list* of releases to be "unlocked". See the -f keyletter for a description of the l flag and the syntax of a *list*.

-a*login* specifies a *login* name, or numerical UNIX System group ID, to be added to the list of users that can make deltas (changes) to the SCCS file. A group ID is equivalent to specifying all *login* names common to that group ID. Several a keyletters can be used on a single admin command line. As many *login*s, or numerical group IDs, as desired can be on the list simultaneously. If the list of users is empty, anyone can add deltas.

-e*login* specifies a *login* name, or numerical group ID, to be erased from the list of users allowed to make deltas (changes) to the SCCS file. Specifying a group ID is equivalent to specifying all *login* names common to that group ID. Several e keyletters can be used on a single admin command line.

-y[*comment*] inserts the *comment* text into the SCCS file as a comment for the initial delta in a manner identical to that of *delta*. Omission of the -y keyletter results in a default comment line being inserted in the form: date and time created *YY/MM/DD HH:MM:SS* by login The -y keyletter is valid only if the -i and/or -n keyletters are specified (i.e., a new SCCS file is being created).

-m[*mrlist*] inserts the list of Modification Requests (*MR*) numbers into the SCCS file as the reason for creating the initial delta in a manner identical to *delta*. The v flag must be set and the *MR* numbers are validated if the v flag has a value (the name of an *MR* number validation program). Diagnostics occur if the v flag is not set or *MR* validation fails.

-h causes admin to check the structure of the SCCS file (see the file format *sccsfile*), and to compare a newly computed check-sum (the sum of all the characters in the SCCS file except those in the first line) with the check-sum that is stored in the first line of the SCCS file. Appropriate error diagnostics are produced.

This keyletter inhibits writing on the file, so that it nullifies the effect of any other keyletters supplied, and is, therefore, only meaningful when processing existing files.

-z recomputes the SCCS file check-sum and stores it in the first line of the SCCS file (see -h, above). Use of this keyletter on a truly corrupted file may prevent future detection of the corruption.

files specifies the files to be processed (see FILES below). If a named file doesn't exist, it is created, and its parameters are initialized according to the specified keyletter arguments. Parameters not initialized by a keyletter argument are assigned a default value. If a named file does

exist, parameters corresponding to specified keyletter arguments are changed, and other parameters are left as is.

If a directory is named, admin behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with "s.") and unreadable files are silently ignored. If a hyphen is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file to be processed. Again, non-SCCS files and unreadable files are silently ignored.

FILES

The last component of all SCCS file names must be of the form s.*filename*. New SCCS files are given mode 444 (see chmod). Write permission in the pertinent directory is, of course, required to create a file. All writing done by admin is to a temporary x-file, called x.*filename*, (see get), created with mode 444 if the admin command is creating a new SCCS file, or with the same mode as the SCCS file if it exists. After successful execution of admin, the SCCS file is removed (if it exists), and the x-file is renamed with the name of the SCCS file. This ensures that changes are made to the SCCS file only if no errors occurred.

It is recommended that directories containing SCCS files be mode 755 and that SCCS files themselves be mode 444. The mode of the directories allows only the owner to modify SCCS files contained in the directories. The mode of the SCCS files prevents any modification at all except by SCCS commands.

If it should be necessary to patch an SCCS file for any reason, the mode can be changed to 644 by the owner allowing use of ed. *Care must be taken*! The edited file should *always* be processed by an admin -h to check for corruption followed by an admin -z to generate a proper check-sum. Another admin -h is recommended to ensure the SCCS file is valid.

Admin also makes use of a transient lock file (called z.*filename*) that is used to prevent simultaneous updates to the SCCS file by different users. See get for further information.

DIAGNOSTICS

- Use help for explanations.

SEE ALSO

sccs, delta, ed, get, help, prs, what
Concentrix Programming Tools Guide: Source Code Control System – System V Interface
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: sccsfile

afio

manipulate archives and files

The afio command is used to manipulate groups of files, copying them within the collective filesystem or between the filesystem and an afio archive.

---

afio -o -t -i [*option...*] *archive*
afio -p [*option...*] *directory* [...]

ARGUMENTS

-o reads pathnames from the standard input and writes an archive.

-t reads an archive and writes a table of contents to the standard output.

-i installs the contents of an archive relative to the working directory.

-p reads pathnames from the standard input and copies the files to each directory.

Creates missing directories as necessary, with permissions to match their parents. Generates sparse filesystem blocks (with lseek(2)) when possible. Supports multi-volume archives during interactive operation (i.e., when /dev/tty is accessible and SIGINT is not being ignored).

DESCRIPTION

Afio manipulates groups of files, copying them within the (collective) filesystem or between the filesystem and an afio archive. Note that afio archives are portable, as they contain only ASCII-formatted header information.

Options

-b size Read or write size-character archive blocks. Suffices of *b*, *k* and *m* denote multiples of 512, 1024 and 1048576, respectively. Defaults to 5120.

-c count Buffer count archive blocks between I/O operations. A large count is recommended with streaming magnetic tape drives.

-d Don't create missing directories.

-e bound Pad the archive to a multiple of bound characters. Recognizes the same suffices as *-s*. Defaults to 1x (the *-b* block size)

-f Spawn a child process to actually write to the archive; provides a clumsy form of double buffering. Requires *-s* for multi-volume archive support.

-g Change to input file directories. Avoids quadratic filesystem behavior with long similar pathnames. Requires all absolute pathnames, including those for the **-o** archive and the **-p** directories.

-h Follow symbolic links, treating them as ordinary files and directories.

-j Don't generate sparse filesystem blocks.

-k Skip corrupt data at the beginning of an archive (rather than complaining about unrecognizable input).

-l With **-o**, write file contents with each hard link.
With **-t**, report hard links.
With **-p**, attempt to link files rather than copying them.

-m Mark output files with a common current timestamp (rather than with input file modification times).

-n Protect newer existing files (comparing file modification times).

-s limit Restrict each portion of a multi-volume archive to limit characters. Recognizes the same suffixes as *-b*. Also, the suffix *x* denotes a multiple of the *-b* block size (and must follow any *-b* specification). Useful with finite-length devices which do not return short counts at end of media; output to magnetic tape typically falls into this category.

-u Report files with unseen links.

-v Verbose. Report pathnames as they are processed. With -t, gives an ls -l style report (including link information).

-x Retain file ownership and setuid/setgid permissions. This is the default for the super-user; he/she may use -X to override it.

-y prefix Restrict archive processing to names beginning with prefix. Specify once for each prefix to be recognized. Use *-Y* to supply prefixes which are not to be processed.

-z Print execution statistics. This is meant for human consumption; use by other programs is officially discouraged.

Special-case archive names:

- Specify [-] to read or write the standard input or output, respectively. This disables multi-volume archive handling.
- Prefix a command string to be executed with an exclamation mark (!). The command is executed once for each archive volume, with its standard input or output piped to afio. It is expected to produce a zero exit code when all is well.
- Use system:file to access an archive in file on system. This is really just a special case of pipelining. It requires a 4.2BSD-style remote shell (rsh) and a remote copy of afio.
- Anything else specifies a local file or device. Creates an output file if it does not already exist.

Recovers from archive corruption by searching for a valid magic number. This is rather simplistic, but, much like a disassembler, almost always works.

Optimizes pathnames with respect to the current and parent directories. For example, ./src/sh/../misc/afio.c becomes src/misc/afio.c.

EXAMPLES

Be sure to use option *-j* on any command lines when extracting files from the afio archive:

```
...extracting Concentrix afio image from /dev/rschs
        cd /tmp; afio -b 20b -ijx -z /dev/rschs
```

The following two examples compare the use of the -y and the -Y options.

The first command will extract *only* /bin/sh and /etc/init from the tape.

```
afio -i -y bin/sh -y etc/init afio_archive
```

The second command will extract all files *except* those in the bin directory. You can see how this works by using the -t option (table of contents); only those entries which would be extracted are printed.

```
afio -i -Y bin afio_archive
```

NOTES

- Restricts pathnames to 1023 characters and 255 meaningful elements.
- There is no sequence information within multi-volume archives. Input sequence errors generally masquerade as data corruption.
- Degenerate uses of symbolic links are mangled by pathname optimization. For example, assuming that usr.src is a symbolic link to /usr/src, the pathname usr.src/../bin/cu is mis-optimized into bin/cu (rather than /usr/bin/cu).

SEE ALSO

bfio, find, tar

alias

bind name to string (C shell)

The alias command binds a single symbolic name to one or more words. The symbolic name can be used as a command; thus, alias allows you to change the names of existing commands and to create new ones.

---

alias [*name* [*wordlist*]]

ARGUMENTS

If no arguments are supplied, alias displays a list of all existing aliases.

name displays the alias for the specified name.

wordlist assigns the specified word list as the alias for *name*.

EXAMPLE

The following alias command assigns the word backup to a remote dump command that might otherwise be difficult to remember.

```
alias backup rdump 1uf DFSLOW1:/dev/rmt8 /dev/sd0g
```

NOTES

- The unalias command removes aliases.
- You cannot use the word `alias` or `unalias` as a name.

SEE ALSO

unalias

alloc

show amount of dynamic core in use (C shell)

The alloc command shows the amount of dynamic core in use, broken down into used and free core, and the address of the last location in the heap.

---

alloc [*arg*]

ARGUMENTS

arg (any string) shows each used and free block on the internal dynamic memory chain indicating its address, size, and whether it is used or free.

EXAMPLE

```
% alloc
11304 used, 19412 free, 4257452 end

% alloc foobar
20077260     20 FREE
20077304     16 BUSY
20077324     12 BUSY
20077340     20 BUSY
     .         .  .
     .         .  .
     .         .  .

20140760     12 BUSY
20140774     32 BUSY
20141034     12 BUSY
20141050      8 BUSY
20141060 13436 FREE
11848 used, 18868 free, 4257452 end
```

apply

apply a command to a set of arguments

The apply command executes a specified command on each of one or more arguments in turn.

---

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

ARGUMENTS

-ac	specifies a substitution character (default is per cent sign (%)).
-*n*	specifies the number of arguments to be passed to *command*. If *n* is zero, *command* runs once for each *arg*.
command	is the command to run.
arg	is an argument on which to apply the command.

DESCRIPTION

Character sequences of the form *%d* in *command*, where *d* is a digit from 1 to 9, are replaced by the *d*'th following unused *arg*. If any such sequences occur, *n* is ignored, and the number of arguments passed to *command* is the maximum value of *d* in *command*.

EXAMPLE

`apply echo *`	is similar to *ls*.
`apply -2 cmp a1 b1 a2 b2 ...`	compares the a files to the b files.
`apply -0 who 1 2 3 4 5`	runs *who* five times.
`apply 'ln %1 /usr/joe' *`	links all files in the current directory to `/usr/joe`.

NOTES

- Shell metacharacters in a command may have bizarre effects; it is best to enclose complicated commands in single quotes.
- There is no way to pass a literal %2 if per cent sign is the argument expansion character.
- If your `.cshrc` file defines the shell variable `prompt`, apply prints its value once for each *arg*.

SEE ALSO

sh

apropos

find pertinent on-line documentation

The apropos command displays a one-line synopsis of each on-line documentation file whose name or description contains a specified word. Once you know what you are looking for, use man to read the appropriate files.

---

apropos *word...*

ARGUMENTS

word is considered separately from each other *word*. The case of letters is ignored.

EXAMPLE

Try `apropos editor` and `apropos password` to get a feel for what apropos does.

NOTES

- Apropos performs a simple string search. For example, `apropos man` finds all instances of `command`, `manipulate`, `mantissa`, `permanent`, and so forth.
- Apropos is really just a symbolic link to man that invokes the –k option.

FILES

`/usr/man/whatis` on-line documentation database file

SEE ALSO

man, whatis, whatisin, whereis
GNU EMACS Manual: Self-Documenting Commands

ar

archive and library maintainer

The ar command maintains groups of files combined into a single archive file. Its main use is to create and update library files as used by the loader. It can be used, though, for any similar purpose.

> NOTE: This version of ar uses a ASCII-format archive that is portable. Programs for dealing with older formats are available: see arcv (*Concentrix-2800 System Administration*).

---

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

ARGUMENTS

key is one *key* character optionally concatenated with one or more modifier characters, as shown below.

posname specifies where the files are to be moved (used with r and m *keys*).

afile specifies the archive file.

name specifies a constituent file in the archive file.

KEYS

d deletes the named files from the archive file.

r replaces the named files in the archive file. By default, new files are placed at the end of the file. If a positioning character is present, the *posname* argument must be present and specifies where the files are to be moved.

q quickly appends the named files to the end of the archive file. Optional positioning characters are invalid. The command does not check whether the added members are already in the archive. Useful only to avoid quadratic behavior when creating a large archive piece-by-piece.

t displays a table of contents of the archive file. If no names are given, all files in the archive are tabled. If names are given, only those files are tabled.

p displays the named files in the archive.

m moves the named files. By default, **m** moves files to the end of the archive. If a positioning character is present, the *posname* argument must be present and specifies where the files are to be moved.

x extracts the named files. If no names are given, all files in the archive are extracted. The archive file is not altered.

MODIFIERS

v gives a file-by-file description of the making of a new archive file from the old archive and the constituent files. When used with t, it gives a long listing of all information about the files. When used with p, it precedes each file with a name.

u when used with r, replaces only those files with 'last-modified' dates later than the archive files.

a when used with r or m, places files after *posname*, which must be present.

i when used with r or m, places files before *posname*, which must be present.

b when used with r or m, places files before *posname*, which must be present.

c suppresses the normal message that is produced when *afile* is created. Normally ar creates *afile* when it needs to.

l causes temporary files to be placed in the local directory. Normally ar places temporary files in the directory `/tmp`.

o when used with x, the 'last-modified' date is reset to the date recorded in the archive. Normally the 'last-modified' date of each extracted file is the date when it is extracted.

NOTES

- The archiver now uses System V library format to maintain all new libraries. Longnames are supported and the symbol index is now part of the archive. Old libraries are still maintained in the 4.2BSD format, and the archiver continues to accept archive files in the old format. If libraries are copied with the new archive format, it is not necessary to run ranlib on them. It is still necessary to do so on archives copied with the old format.
- If the same file is mentioned twice in an argument list, it can be put in the archive twice.
- The 'last-modified' date of a file is not altered by the o option if the user is not the owner of the extracted file, or the super-user.
- Before building an archive, use the lorder and tsort commands to order the object files so that ld can process the archive file faster. For example:

```
% ar cr library.a `lorder *.o | tsort`
% ranlib library.a
```

FILES

`/tmp/v*` temporaries

SEE ALSO

lorder, ld, ranlib
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: ar
Concentrix-2800 System Administration: arcv

as

Alliant FX/2800 Series assembler

The as command assembles the named file. The output of the assembly is an object file, which must be linked by the *ld* command to produce an executable program.

---

as [-l] [-I*directory*]... [-M] [-o *objfile*] [-D*name=value*] [*sourcefile*]

ARGUMENTS

-l produces an assembly listing on the standard output.

-I*directory* causes the directory named to be added to the search list for include files. When the assembler encounters an include file, it first searches the current directory, them searches the directories specified by the -I options.

-M causes the assembler to output (on the standard output) a dependency list that can be used in make files. The -M option overrides the -l option; no listing file is produced.

-o *objfile* explicitly names the object file. The default is the name of the source file with an extension of .o. If the source file name has an extension of .s, .o replaces it; otherwise, .o is added to the name. Note that the -o option uses a space before its argument.

-D*name=value*
defines the specified symbol, *name*, and assigns it *value*.

sourcefile specifies a source file. By convention, a source file has an extension of .s. The default is the standard input. However, if you default the source file to standard input, you must specify the -o option to produce an object file.

EXAMPLES

The following example assembles code from a source file named sum.s:

```
% as sum.s
```

The following example assembles, links, and executes code from the source file sum.s, producing an object file named sum.o and an executable file named a.out:

```
% as sum.s
% ld sum.o
% a.out
```

The following example assembles two source files, sum.s and sum2.s, then links them as one executable file. The object files are named sum.o (by default) and sumjr.o (explicitly). The executable file is named sum.

```
% as sum.s
% as -o sumjr.o sum2.s
% ld -o sum sum.o sumjr.o
```

The following example produces a listing file named `sum.lis`:

```
% as -l sum.s > sum.lis
```

The following example assigns the value 3 to the symbol `sym_1`:

```
% as -Dsym_1=3 vsum.s
```

FILES

name.o	default resultant object file

SEE ALSO

ld, adb,
Concentrix-2800 Systems Programming Guide: Assembler

at

execute commands at a later time

Submits a job to execute a shell script at a specified later time.

---

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

ARGUMENTS

-c uses the C shell (defaults to the current environment shell).

-s uses the Bourne shell (defaults to the current environment shell).

-m sends mail to the user after the job completes. If job ends in an error, the mail contains the error diagnostics. If the job is successful, the mail states that no errors occurred.

time one to four digits, with an optional one-character suffix:

a	AM
p	PM
n	noon
m	midnight

One- and two-digit numbers specify hours; three- and four-digit numbers specify hours and minutes. If the suffix is omitted, a 24-hour clock is assumed.

day either a month name followed by a day number, or a day of the week; if the keyword **week** follows, invocation is moved seven days further off. Names of months and days can be recognizably truncated.

file a shell script. Defaults to standard input.

DESCRIPTION

Jobs are executed by periodic execution of the program /usr/lib/atrun through the cron command. The granularity of at depends on how often atrun is executed.

At execution time, an at job changes directory to the user's directory at job submission, sets the umask to the user's umask at job submission, sets all environment variables (except the variable TERM, which is useless in this context) to the user values at job submission, and executes the script. The job uses the user and group IDs of the user who submitted the job.

EXAMPLES

The following example submits a job:

```
% cat script
#
# Sample script for "at"
#
ls -l > outputfile          Note redirection - stdout is null.
% at -m 1930 script         Run script at 7:30 p.m. and return mail.
%                           Job is now queued.
```

The following example shows how to submit a job that you type in:

```
% at -m 1930
ls -l >outputfile          Same as above.
CTRL D                     Ends typed-in commands.
%                          Job is now queued.
```

Examples of valid command lines are as follows:

```
% at 9a dec 15 script
% at 1530 friday week script
```

DIAGNOSTICS

- Syntax errors and times that are out of range.

NOTES

- A file that is not a shell script usually gets queued but does not run as expected.
- If the system crashes while a job is running, mail is not sent to the user telling him or her that the job failed to complete.
- Standard output is lost unless redirected. Error output is lost unless redirected or -m is specified.
- Due to the granularity of the execution of /usr/lib/atrun, scheduling almost exactly 24 hours into the future may not work properly.
- The directory /usr/spool/at/past contains jobs whose execution time passed without execution because of system shutdown, date changes by a system operator, or other chronological anomalies. The file should be examined and purged periodically.

FILES

/usr/lib/atrun	job executor (run by cron)
/usr/spool/at/yy.ddd.hhhh.*	jobs to be executed
/usr/spool/at/lasttimedone	last time atrun ran a job
/usr/spool/at/past	scripts whose execution times have passed
/usr/spool/at	spooling area

SEE ALSO

atq, atrm, calendar, sleep
Concentrix-2800 System Administration: cron

atq

print the queue of jobs waiting to be run

Prints the following information about at jobs awaiting execution: rank (precedence in the job queue), execution date, owner, job number, and job name (name of the submitted script or stdin). Also prints the last time an at job executed.

---

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

ARGUMENTS

-c sorts the display by time of entry, earliest job first. By default, jobs are sorted by job precedence (execution time, next job first).

-n prints only the total number of jobs in the at queue.

user-name
prints only the jobs associated with the specified user.

EXAMPLES

Note the following example:

```
% atq

 LAST EXECUTION TIME: Mar 24, 1988 at 16:30

 Rank     Execution Date     Owner      Job #   Job Name
  1st   Mar 24, 1988 19:00   hacker    12663   AT
  2nd   Mar 24, 1988 20:00   hacker    12665   AT
```

atrm

remove jobs queued by at

Removes jobs from the at queue.

---

at [-f] [-i] [-] [*job-no*,...] [*user-name*,...]

ARGUMENTS

-f suppresses the display `job-no: removed` for each job removed.

-i interactively inquires **remove it?** for each job to be removed. A response of `y` or `Y` removes the job; any other response retains the job.

- removes all jobs owned by the user issuing the command.

user-name
(only useful for the superuser) removes *all* jobs associated with the specified user(s).

job-no removes the specified job(s). Does not apply if *user-name* is specified.

EXAMPLES

Note the following examples:

```
% atrm 12663
 12663: removed

% atrm 12663 12665
 12663: removed
 12665: removed

% atrm -f 12663 12665

% atrm -
 12663: removed
 12665: removed

% atrm -i -
 12663: remove it? y
 12665: remove it? n

# atrm -f hacker
```

SEE ALSO

at, atq

at_sign

set local numeric variable (C shell)

The @ command evaluates a numeric expression and assigns the result (in the form of a numeric string) to the specified variable.

> **NOTE:** The following synopsis contains some brackets that are literals and some that are meta-syntax.

@

@ *name* = *expr*

@ *name*[*index*] [=*expr*]

ARGUMENTS

With no arguments, @ prints the values of all shell variables.

name is the name of the variable to set. Variable names consist of 1 to 20 characters chosen from the same set of characters as a filename (letters, underscore, and digits, beginning with a letter). The space separating *name* from = is optional.

name[*index*] specifies an element of *name*. To access individual elements of an array, use *index* numbers (counting from one). To access a range of elements, use a pair of *index* numbers separated with a hyphen. If omitted, the second *index* number defaults to the last element.

expr is a numeric expression as specified in the *Concentrix System User's Guide, C Shell Reference*. Spaces are mandatory in separating components of *expr* which would otherwise be single words.

DESCRIPTION

Shell variables can be used in numeric calculations. Strings of digits are treated as integers (the null string is treated as zero). Strings containing non-numeric characters are invalid. The second and subsequent words of multiword values are ignored.

All the arithmetic operators of the C language are available with the same precedence. In particular, == and != compare strings and && and || perform boolean and/or operations.

The special operators =~ and !~ are similar to == and != except that the string on the right side can have pattern matching characters (like asterisk, question mark, or brackets) and test whether the string on the left matches the pattern on the right.

Postfix operators (++ and --) increment and decrement *name* respectively, i.e. @ i++. If *expr* contains <, >, & or |, that part of the expression must be placed within parentheses.

EXAMPLE

```
% @ x = 2
% @ y = 3
% @ z = $x * $y
% echo $z
6
```

SEE ALSO

set

autopsy

utility to examine crash dumps

autopsy is a program that reads a Concentrix-2800 crash dump from tape or disk, and allows the saved operating system state to be examined.

---

autopsy [–r] [–s *seg*] [–t [*tape*]] [*file*]

ARGUMENTS

–r	include the pages from the ram disk.
–s *seg*	use *seg* to override the segment table address.
–t *tape*	creates a dump file on disk (default: os.core) from the specified dump tape (default: /dev/rschs). Autopsy reads the input file, prints some statistics, and writes the pages important to the operating system to the os.core file.
file	specifies the crash dump file to examine (default: os.core).

DESCRIPTION

Autopsy has commands to read program memory, display register contents, decode C stack traces, do integer arithmetic, handle instruction typeout, and interpret bit patterns in a variety of ways. Autopsy does not allow memory to be changed and does not support commands related to kernel execution.

When autopsy is ready for you to start analyzing, it selects a processor and prints a list of all the processors in the machine, their PCs, and the panicstr. If there is unprinted data in the printf buffer, that is printed.

To read from the default tape to a different file, use the following syntax:

```
% autopsy <file> -t [-r]
```

To exit from autopsy, use the :e or :q command or type CTRL D.

NOTE: Dump tapes can be copied to disk via the Concentrix-2800 dd command. The disk file is then suitable for autopsy –t but takes up more disk space.

For a full description of autopsy (and the related kernel debugger, ddt) see *Concentrix-2800 Systems Programming Guide*.

SEE ALSO

Concentrix-2800 Systems Programming Guide, autopsy/ddt

awk

pattern scanning and processing language

The awk command scans files for lines that match any of a set of patterns. With each pattern, there can be an associated action that is performed each time a match occurs.

---

awk [-Fc] [*pattern*] [*file...*]

awk [-Fc] [-f *patfile*] [*file...*]

ARGUMENTS

-Fc specifies a one-character field separator (see notes).

pattern is a quoted string containing a set of patterns. For example, to search `myfile` for lines longer than 72 characters.

```
% awk 'length > 72' myfile
```

-f *patfile* specifies a file containing a set of patterns.

file specifies a file to be scanned. A hyphen specifies the standard input (the default).

> NOTE: The case of the option letter "f" is insignificant. The -Fc option can be specified -fc and the -f *file* option also can be specified -F *file*. The meaning of the option depends on whether or not there is a space between the option and its argument.

DESCRIPTION

Files are read in order. Each line is matched against the pattern portion of every pattern-action statement; the associated action is performed for each matched pattern.

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

A pattern-action statement has the form

pattern { *action* }

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

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

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

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

The *print* statement displays its arguments on the standard output (or on a file if >*file* is present), separated by the current output field separator, and terminated by the output record separator. The *printf* statement formats its expression list according to the format (see *Concentrix-2800 System Reference Manual: System Calls and C Library Subroutines* :printf).

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

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

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

A relational expression is one of the following:

expression matchop regular-expression
expression relop expression

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

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

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

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

or by using the −F*c* option.

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

EXAMPLE

Print first two fields in opposite order:

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

Add up first column, print sum and average:

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

Print fields in reverse order:

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

Print all lines between start/stop pairs:

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

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

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

NOTES

- There are no explicit conversions between numbers and strings. To force an expression to be treated as a number add 0 to it; to force it to be treated as a string: concatenate the null string to it.

SEE ALSO

lex, sed
Concentrix Programming Tools Guide: Awk – Pattern Scanning and Processing Language

basename

strip path name prefix and suffix

The basename command returns a specified string stripped of any directory prefix and specific suffix characters. It removes any prefix ending with a slash character and the specified suffix, if present from the specified string. Basename is normally used inside substitution marks in shell procedures.

---

basename *string* [*suffix*]

ARGUMENTS

string specifies a string from which to extract the base name.

suffix specifies a suffix to remove.

EXAMPLE

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

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

SEE ALSO

sh

bc

arbitrary-precision arithmetic language

The bc command invokes an interactive processor for a language that resembles C but provides unlimited precision arithmetic. It takes input from any files given, then reads the standard input.

---

bc [–c] [–l] [*file...*]

ARGUMENTS

–c (compile only) causes *bc* to send *dc* input to the standard output. *Bc* is actually a preprocessor for *dc*, which it invokes automatically, unless the –c option is present.

–l stands for the name of an arbitrary precision math library.

file specifies an input file.

DESCRIPTION

All function arguments are passed by value.

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

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

The syntax for *bc* programs follows.

- L means letter a-z
- E means expression
- S means statement.

Comments

Comments are enclosed in /* and */.

Names

Simple variables L

Array elements L [E]

The words 'ibase', 'obase', and 'scale'.

Other operands

Arbitrarily long numbers with optional sign and decimal point

(E)

sqrt (E)

length (E) number of significant decimal digits

scale (E) number of digits right of decimal point

L (E , ... , E)

Operators

+ - * / % ^ (% is remainder; ^ is power)

++ -- (prefix and postfix; apply to names)

== <= >= != < >

= += -= *= /= %= ^=

Statements

E

{ S ; ... ; S }

if (E) S

while (E) S

for (E ; E ; E) S

null statement

break

quit

Function definitions

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

Functions in -l math library

s(x) sine

c(x) cosine

e(x) exponential

l(x) log

a(x) arctangent

j(n,x) Bessel function

EXAMPLE

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

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

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

displays approximate values of the exponential function of the first ten integers.

NOTES

- No &&, ||, or ! operators.
- *For* statement must have all three E's.
- *Quit* is interpreted when read, not when executed.

FILES

/usr/lib/lib.b	mathematical library
dc	desk calculator proper

SEE ALSO

dc
Concentrix Programming Tools Guide: BC – Arbitrary Precision Desk-Calculator Language

bfio

manipulate archives and files

The bfio command is used to manipulate groups of files, copying them within the collective filesystem or between the filesystem and a bfio archive.

---

bfio -o -t -i [*option...*] *archive*
bfio -p [*option...*] *directory* [...]

ARGUMENTS

-o reads pathnames from the standard input and writes an archive.

-t reads an archive and writes a table of contents to the standard output.

-i installs the contents of an archive relative to the working directory.

-p reads pathnames from the standard input and copies the files to each directory.

Creates missing directories as necessary, with permissions to match their parents. Generates sparse filesystem blocks (with lseek) when possible. Supports multi-volume archives during interactive operation (i.e., when /dev/tty is accessible and SIGINT is not being ignored).

DESCRIPTION

bfio manipulates groups of files, copying them within the (collective) filesystem or between the filesystem and a bfio archive. Note that bfio archives are portable, as they contain only ASCII-formatted header information.

bfio and afio comparison

Bfio creates a tape which includes FX/2800 32bit device numbers; afio does not. For the most part, tapes generated by bfio can be read by afio, but if FX/2800 device entries are backed up, that tape can be read by afio only if the -k modifier is used (which allows skipping what appears to be a corrupt file header).

Similarly, tapes generated by afio can be read by bfio, but device entries written on an FX/80 Series machine are not created correctly on an FX/2800.

Options

-b size Read or write size-character archive blocks. Suffices of *b*, *k* and *m* denote multiples of 512, 1024 and 1048576, respectively. Defaults to 5120.

-c count Buffer count archive blocks between I/O operations. A large count is recommended with streaming magnetic tape drives.

-d Don't create missing directories.

-e bound Pad the archive to a multiple of bound characters. Recognizes the same suffices as *-s*. Defaults to 1x (the *-b* block size)

-f Spawn a child process to actually write to the archive; provides a clumsy form of double buffering. Requires *-s* for multi-volume archive support.

-g Change to input file directories. Avoids quadratic filesystem behavior with long similar pathnames. Requires all absolute pathnames, including those for the **-o** archive and the **-p** directories.

-h Follow symbolic links, treating them as ordinary files and directories.

-j Don't generate sparse filesystem blocks.

-k Skip corrupt data at the beginning of an archive (rather than complaining about unrecognizable input).

-l With **-o**, write file contents with each hard link.
With **-t**, report hard links.
With **-p**, attempt to link files rather than copying them.

-m Mark output files with a common current timestamp (rather than with input file modification times).

-n Protect newer existing files (comparing file modification times).

-s limit Restrict each portion of a multi-volume archive to limit characters. Recognizes the same suffixes as *-b*. Also, the suffix *x* denotes a multiple of the *-b* block size (and must follow any *-b* specification). Useful with finite-length devices which do not return short counts at end of media; output to magnetic tape typically falls into this category.

-u Report files with unseen links.

-v Verbose. Report pathnames as they are processed. With **-t**, gives an ls -l style report (including link information).

-x Retain file ownership and setuid/setgid permissions. This is the default for the super-user; he/she may use -X to override it.

-y prefix Restrict archive processing to names beginning with prefix. Specify once for each prefix to be recognized. Use *-Y* to supply prefixes which are not to be processed.

-z Print execution statistics. This is meant for human consumption; use by other programs is officially discouraged.

Special-case archive names:

- Specify [-] to read or write the standard input or output, respectively. This disables multi-volume archive handling.

- Prefix a command string to be executed with an exclamation mark (!). The command is executed once for each archive volume, with its standard input or output piped to bfio. It is expected to produce a zero exit code when all is well.
- Use system:file to access an archive in file on system. This is really just a special case of pipelining. It requires a 4.2BSD-style remote shell (rsh) and a remote copy of bfio.
- Anything else specifies a local file or device. Creates an output file if it does not already exist.

Recovers from archive corruption by searching for a valid magic number. This is rather simplistic, but, much like a disassembler, almost always works.

Optimizes pathnames with respect to the current and parent directories. For example, ./src/sh/../misc/bfio.c becomes src/misc/bfio.c.

EXAMPLE

Be sure to use option *-j* on any command lines when extracting files from the bfio archive:

```
...extracting Concentrix bfio image from /dev/rschs
        cd /tmp; bfio -b 20b -ijx -z /dev/rschs
```

The following two examples compare the use of the -y and the -Y options.

The first command will extract *only* /bin/sh and /etc/init from the tape.

```
bfio -i -y bin/sh -y etc/init bfio_archive
```

The second command will extract all files *except* those in the bin directory. You can see how this works by using the -t option (table of contents); only those entries which would be extracted are printed.

```
bfio -i -Y bin bfio_archive
```

NOTES

- Restricts pathnames to 1023 characters and 255 meaningful elements.
- There is no sequence information within multi-volume archives. Input sequence errors generally masquerade as data corruption.
- Degenerate uses of symbolic links are mangled by pathname optimization. For example, assuming that usr.src is a symbolic link to /usr/src, the pathname usr.src/../bin/cu is mis-optimized into bin/cu (rather than /usr/bin/cu).

SEE ALSO

afio, find, tar

bg

execute a job in the background (C shell)

The bg command puts a job into the background and, if stopped, restarts it.

---

bg [*%job*]...

ARGUMENTS

%job specifies the number of the job to put into the background.

EXAMPLE

The following command sequence:

```
% nroff myfile
CTRL Z
Stopped
% bg
```

has the same effect as if you had originally typed:

```
% nroff myfile&
```

SEE ALSO

fg, jobs, percent_sign

biff

be notified if mail arrives

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

---

biff y

biff n

DESCRIPTION

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

Biff operates asynchronously. For synchronous notification use the MAIL variable of sh or the *mail* variable of csh.

SEE ALSO

csh, sh, mail
Concentrix-2800 System Administration: comsat

binmail

send or receive mail among users

Mail with no argument prints a user's mail, message-by-message, in last-in, first-out order; the optional argument + displays the mail messages in first-in, first-out order. For each message, mail reads a line from the standard input to direct disposition of the message.

Note: This is the old version 7 UNIX system mail program. The default mail command is described under mail, and its executable file is in the directory /usr/ucb.

/bin/mail [+] [–i] [*person*] ...

/bin/mail [+] [–i] –f *file*

COMMANDS

RETURN	go on to next message.
d	delete message and go on to the next.
p	print message again.
–	go back to previous message.
s [*file*] ...	save the message in the named files (mbox default).
w [*file*] ...	save the message, without a header, in the named files (mbox default).
m [*person*] ...	mail the message to the named persons (you are the default).
CTRL D	put unexamined mail back in the mailbox and stop.
q	same as EOT.
!*command*	escape to the shell and execute the specified command.
*	print a command summary.

An interrupt normally terminates the mail command; the mail file is unchanged. The optional argument –i tells mail to continue after interrupts.

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

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

A user is informed of the presence of mail at login.

FILES

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

SEE ALSO

mail, write, uucp, uux, xsend)
Concentrix-2800 System Administation: sendmail

BUGS

Race conditions sometimes result in a failure to remove a lock file.

Normally anybody can read your mail, unless it is sent by xsend. An installation can overcome this by making mail a set-user-id command that owns the mail directory.

break

resume execution after foreach or while (C shell)

The break command is intended for use in shell scripts. It causes execution to resume after the *end* of the nearest enclosing *foreach* or *while*. The remaining commands on the current line are executed. Multi-level breaks are thus possible by writing them all on one line.

---

break

SEE ALSO

breaksw, foreach, switch, while

break

exit from enclosing for or while loop (Bourne shell)

The break command exits from the enclosing *for* or *while* loop. It accepts an optional numeric argument that specifies how many levels to exit.

---

break [*n*]

ARGUMENTS

n	If *n* is specified, break *n* levels.

SEE ALSO

for, switch, while

breaksw

break from switch command (C shell)

The breaksw command is intended for use in shell scripts. It causes a break from a switch command, resuming after the endsw.

---

breaksw

SEE ALSO

break, switch

cal

display calendar

The cal command displays a calendar for the specified year.

---

cal [*month*] *year*

ARGUMENTS

month is a number between 1 and 12. If specified, a calendar just for that month is displayed.

year is a number between 1 and 9999. Thus, `cal 78` refers to the early Christian era, not the 20th century.

EXAMPLE

```
% cal 1 2000

January 2000
 S  M Tu  W Th  F  S
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

NOTES

- The year is always considered to start in January.

calendar

reminder service

The calendar command consults the file `calendar` in the current directory and displays lines that contain today's or tomorrow's date anywhere in the line.

---

calendar [-]

ARGUMENTS

- causes calendar to run for every user who has a file `calendar` in his login directory and send any positive results by mail. Normally this is done daily under control of cron *(Concentrix-2800 System Administation)*.

DESCRIPTION

Most reasonable month-day dates such as `Dec. 7`, `december 7`, `12/7`, etc., are recognized, but not `7 December` or `7/12`. If you give the month as an asterisk with a date, i.e. `* 1`, that day in any month will do. On weekends, `tomorrow` extends through Monday.

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

NOTES

- Calendar's extended idea of tomorrow doesn't account for holidays.

FILES

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

SEE ALSO

at, mail
Concentrix-2800 System Administation: cron

case

denote label in switch command (C shell)

The C-shell **case** command is intended for use in C-shell scripts, as a part of the switch command. **Case** denotes a label in a **switch** command.

---

case *label*:

See the **switch** command for full details.

SEE ALSO

switch

case

control multi-way branching in shell scripts (Bourne shell)

The Bourne shell **case** command allows for multi-way branching in a shell script. It performs the same function as the C shell switch command.

---

case *word* **in**
  label) *list* ;;
  label) *list* ;;
  label) *list* ;;
esac

ARGUMENTS

word — a string that is used to determine where the **case** command branches to. While it is not required, *word* is almost always a string variable.

label — a string that is used to mark a branch location. A *label* may be either a literal (non-variable) string, or a string pattern.

list — a list of commands to be executed.

DESCRIPTION

The Bourne shell **case** command implements standard multi-way branching for the Bourne shell.

At the time the **case** command is executed, the shell evaluates the string in *word*. The shell then compares *word* with each of the strings in the set of *labels*. The shell passes control to the first *label* that matches *word*. After the commands following the label are executed, control passes to the statement following **esac**.

The order of the *labels* is important, since control is passed to the first *label* that matches *word*.

EXAMPLE

This example shows how a string variable is used for *word*, and that the order of the *labels* is important.

```
case $1 in
  *.doc) print $1 ;;
  *.c) cc $1 ; print $1 ;;
  *) echo Unknown file type.
esac
```

Suppose the name of this shell script is **do_the_right_thing**.

Typing "**do_the_right_thing acct_pay.doc**" causes the Bourne shell to print the file **acct_pay.doc**. Typing "**do_the_right_thing acct_pay.c**" causes the Bourne shell to compile the program **acct_pay.c**, and to print its source code. Typing "**do_the_right_thing acct_pay.foo**" causes the Bourne shell to display the error message "**Unknown file type.**".

Note that even though any string in *word* matches the third *label*, the third *label* is only executed if *word* does not match either of the first two *labels*.

cat

display text file

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

---

cat [-u] [-n] [-b] [-s] [-v] [-e] [-t] *file...*

ARGUMENTS

-u makes the output completely unbuffered.

By default, the output is buffered in blocks of the size recommended by stat(2); unless the standard output is a terminal, in which case the output is line buffered.

-n displays the output lines preceded by line numbers, numbered sequentially from one.

-b used with -n omits the line numbers from blank lines.

-s crushes out multiple adjacent empty lines so that the output is displayed single spaced.

-v displays non-printing characters so that they are visible. Control characters appear as ^X for CTRL X; the delete character (octal 0177) appears as ^?. Non-ASCII characters (with the high bit set) are displayed as M- (for meta) followed by the character of the low 7 bits.

-e used with -v displays a dollar sign character at the end of each line.

-t used with -v displays tab characters as `^I`.

file specifies an input file. A hyphen specifies the standard input (the default).

EXAMPLE

```
cat file
```

displays the file on the standard output.

```
cat file1 file2 >file3
```

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

NOTES

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

SEE ALSO

cp, ex, more, pr, tail

cb

C program beautifier

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

---

cb

---

cc

C language compiler

The cc command is a link that invokes one of the following C compilers:

The cc command invokes either the FX/C–2800 compiler (as if invoked by the fxc command), or the Scalar/C compiler (as if invoked by the scc command).

It is best to use the fxc or scc command to call the appropriate compiler; however, if you want to use the cc command, and are unsure which compiler it calls, you can query the compiler for its name and version by typing `cc -v`.

SEE ALSO

scc
FX/2800 C Programmers's Handbook: fxc
Concentrix-2800 Scalar/C Programmer's Handbook: scc

ccat

display compressed files (old command, found in usr/old, replaced by zcat)

The ccat command cats the original file from a file compressed by compact, without uncompressing the file.

---

ccat [*file...*]

RESTRICTION

The last segment of the filename must contain fewer than 13 characters to allow space for the appended `.C`.

FILES

`*.C`	compacted file created by compact, removed by uncompact

SEE ALSO

cat, compact, uncompact,
Variations on a Theme of Huffman, Gallager, Robert G.,
I.E.E.E. Transactions on Information Theory, vol. IT-24, no. 6,
November 1978, pp. 668 - 674.

cd

change working directory(C shell)

The cd command makes the specified directory your new working directory. You must have execute (search) permission in the specified directory.

---

cd [*name*]

ARGUMENTS

name specifies the new working directory. The default is your login directory (~).

DESCRIPTION

Cd is built into the C shell. Because a new process is created to execute each command, cd would be ineffective if it were written as a normal command.

If you are using the C shell, and *name* is not found as a subdirectory of the current directory (and does not begin with slash, dot-slash, or dot-dot-slash), each component of the environment variable cdpath is checked to see if it has a subdirectory *name*. If that fails, but *name* is a shell variable whose value begins with slash, it is tried to see if it is a directory.

NOTES

- Cd refers to the shell variable home.
- Cd has the alternate form chdir.
- The C shell's cd command with an argument does not work properly once you have crossed through a symbolic link.

SEE ALSO

chdir, csh, sh, pwd,
Concentrix-2800 System Reference Manual: System Calls and C Library Subroutines: chdir

cd

change working directory(Bourne shell)

The cd command makes the specified directory your new working directory. You must have execute (search) permission in the specified directory.

---

cd [*name*]

ARGUMENTS

name specifies the new working directory. The default is your login directory (~).

DESCRIPTION

Cd is built into the the Bourne shell. Because a new process is created to execute each command, *cd* would be ineffective if it were written as a normal command.

NOTES

- The shell parameter $HOME is the default for *name*.
- Cd has the alternate form chdir.

SEE ALSO

chdir, csh, sh, pwd,
Concentrix-2800 System Reference Manual: System Calls and C Library Subroutines: chdir

cdc

change the delta commentary of an SCCS delta

The cdc command changes the delta *commentary*, for the *SID* specified by the -r keyletter, of each named SCCS file. Delta *commentary* is defined to be the Modification Request (MR) and comment information normally specified via the delta command (-m and -y keyletters).

---

cdc -r*SID* [-m[*mrlist*]] [-y[*comment*]] *file...*

ARGUMENTS

Arguments to cdc, which can appear in any order, consist of *keyletter* arguments, and file names. All the described *keyletter* arguments apply independently to each named file:

-r*SID* specifies the SCCS *ID*entification (*SID*) string of a delta for which the delta commentary is to be changed.

-m[*mrlist*] if the SCCS file has the v flag set (see admin), a list of MR numbers to be added and/or deleted in the delta commentary of the *SID* specified by the -r keyletter *can* be supplied. A null MR list has no effect.

MR entries are added to the list of MRs in the same manner as that of delta. In order to delete an MR, precede the MR number with the character ! (see *EXAMPLES*). If the MR to be deleted is currently in the list of MRs, it is removed and changed into a "comment" line. A list of all deleted MRs is placed in the comment section of the delta commentary and preceded by a comment line stating that they were deleted.

If -m is not used and the standard input is a terminal, the prompt MRs? is issued on the standard output before the standard input is read; if the standard input is not a terminal, no prompt is issued. The MRs? prompt always precedes the comments? prompt (see -y keyletter).

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

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

-y[*comment*]
specifies arbitrary text used to replace the *comment*(s) already existing for the delta specified by the -r keyletter. The previous comments are kept and preceded by a comment line stating that they were changed. A null *comment* has no effect.

If -y is not specified and the standard input is a terminal, the prompt comments? is issued on the standard output before the standard input is read; if the standard input is not a terminal, no prompt is issued. An unescaped new-line character terminates the *comment* text.

files specifies the files to be processed. If a directory is named, *cdc* behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored. If a name of "-" is given, the standard input is read (see *WARNINGS*); each line of the standard input is taken to be the name of an SCCS file to be processed.

DESCRIPTION

The exact permissions necessary to modify the SCCS file are documented in the *Concentrix Concentrix Programming Tools Guide: Source Code Control System - System V Interface*. Simply stated:

- If you made the delta, you can change its delta commentary; or
- If you own the file and directory you can modify the delta commentary.

EXAMPLE

The following command adds bl78-12345 and bl79-00001 to the MR list, removes bl77-54321 from the MR list, and adds the comment trouble to delta 1.6 of s.file.

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

The following command does the same thing.

```
cdc -r1.6 s.file
MRs? !bl77-54321 bl78-12345 bl79-00001
comments? trouble
```

WARNINGS

- If SCCS file names are supplied to the cdc command via the standard input (- on the command line), the -m and -y keyletters must also be used.

DIAGNOSTICS

- Use *help* for explanations.

FILES

x-file	(see delta)
z-file	(see delta)

SEE ALSO

sccs, admin, delta, get, help, prs
Concentrix Programming Tools Guide: Source Code Control System - System V Interface
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: sccsfile

chdir

change working directory

The chdir *(cd)* command makes the specified directory your new working directory. You must have execute (search) permission in the specified directory.

---

chdir [*name*]

ARGUMENTS

name specifies the new working directory. The default is your login directory (-).

DESCRIPTION

Chdir is built into the C shell and the Bourne shell. Because a new process is created to execute each command, chdir would be ineffective if it were written as a normal command.

If you are using the C shell, and *name* is not found as a subdirectory of the current directory (and does not begin with slash, dot-slash, or dot-dot-slash), each component of the environment variable `cdpath` is checked to see if it has a subdirectory *name*. If that fails, but *name* is a shell variable whose value begins with slash, it is tried to see if it is a directory.

NOTES

- Chdir refers to the shell variable `HOME` (Bourne shell) or `home` (C shell).
- The C shell's `chdir` command with an argument does not work properly once you have crossed through a symbolic link.

SEE ALSO

cd, csh, sh, pwd,
Concentrix-2800 System Reference Manual: System Calls and C Library Subroutines: chdir

checkeq

error checker for mathematics typesetting (unsupported)

The checkeq command checks eqn and neqn files for errors. It reports missing or unbalanced delimiters and `.EQ`/`.EN` pairs.

---

checkeq [*file...*]

ARGUMENTS

file is the name of a file to check for errors.

SEE ALSO

eqn, neqn, troff, tbl
Document Preparation Guide: EQN Mathematics Typesetting
Document Preparation Guide: Using NROFF/TROFF

checknr

error checker for nroff/troff (unsupported)

The checknr command checks a list of nroff or troff input files for certain kinds of errors involving mismatched opening and closing delimiters and unknown commands. Checknr knows about the *ms* and *me* macro packages.

---

checknr [-s] [-f] [-a] [-c] [*file*...]

ARGUMENTS

-s ignores \s size changes.

-f ignores \f font changes.

-a adds additional pairs of macros to the list. This option must be followed by groups of six characters in the form:

```
xx.yy
```

where each group defines a pair of macros .*xx* and .yy. For example, to define a pair .BS and .ES:

```
checknr -a.BS.ES
```

-c defines additional commands. This option must be followed by groups of three characters in the form:

```
.xx
```

where each group defines a command .xx. For example, to define the commands .AA, .BB, and .CC:

```
checknr -c.AA.BB.CC
```

file specifies an input file. The standard input is read by default.

DESCRIPTION

Delimeters checked are:

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

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

DIAGNOSTICS

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

NOTES

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

SEE ALSO

nroff, troff, checkeq
Document Preparation Guide: –ms Macros, –me Macros

chfn

change finger entry

The chfn command is now a part of passwd, by using the passwd –f option.

chgrp

change file group

The chgrp command changes the group-ID of one or more files.

---

chgrp [–f] [–R] *group file...*

ARGUMENTS

–f suppresses error reporting.

–R recursively descends directories it encounters, setting the group–ID. When symbolic links are encountered, their group is changed, but the link is not traversed.

group can be either a decimal GID or a group name found in the group-ID file.

file... a list of files and directories.

DESCRIPTION

The user invoking chgrp must belong to the specified group and be the owner of the file, or be the super-user.

FILES

`/etc/group`

SEE ALSO

Concentrix-2800 System Reference: System Calls and C Library Subroutines: chown
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: passwd, group

chkey

change your encryption key

The chkey command prompts for the user's login password, and uses it to encrypt a new encryption key for the user to be stored in the publickey database.

---

chkey

SEE ALSO

keylogin
Concentrix-2800 System Administation: keyserv, newkey
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: publickey
Concentrix Network File System User's Guide, chapter on NFS administration

chmod

change file access mode

The chmod command changes the access mode (permissions) of one or more files. Use the `ls -l` command to examine access modes.

---

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

ARGUMENTS

-f do not complain if chmod fails to set the mode on a file.

-R recursively descend the directory arguments, setting the mode for each file found. When a symbolic link is encountered, its mode is not changed and it is not traversed.

mode can be symbolic or absolute, as described below.

file... a list of files and/or directories.

Symbolic Modes

A symbolic *mode* has the form:

[who] op permission [op permission]...

who is a combination of the letters:

u	user
g	group
o	other
a	all (equivalent to ugo)

The presence of *who* indicates that *permission* is to be taken from the current mode. If *who* is omitted, the default is **a** but the setting of the file creation mask (see *umask*) is taken into account.

op is one of the following operators:

+	adds *permission* to the file's mode.
-	removes *permission*.
=	assigns *permission* absolutely (all other bits are reset).

permission is any combination of the letters:

r	read
w	write

x	execute
X	set execute only if file is a directory or some other execute bit is already set
s	set owner or group id to file owner on execution
t	save text – sticky

Omitting *permission* is only useful with the *op* = to remove all permissions.

Absolute Modes

An absolute *mode* is an octal number that represents a 12-bit binary file access code as shown below:

bit	who	permission
0	others	execute
1	others	write
2	others	read
3	group	execute
4	group	write
5	group	read
6	owner	execute
7	owner	write
8	owner	read
9	sticky (see the system call *chmod*)	
10	set group ID on execution	
11	set user ID on execution	

NOTES

- Multiple symbolic modes separated by commas can be given.
- Operations are performed in the order specified.
- The permission s (set owner id or group id to file owner on execution) is only useful with u (user) or g (group).
- Only the owner of a file (or the super-user) can change its mode.

EXAMPLE

`chmod o-w file`	denies write permission to others
`chmod +X file`	makes the file executable by all if it is executable by anyone
`chmod 755 file`	no write access for group or others

SEE ALSO

ls
Concentrix-2800 System Administation: chown
Concentrix-2800 System Reference Manual: System Calls and C Library Subroutines: chmod, stat, umask
Concentrix System User's Guide: The File System

chsh

change login shell

The chsh command is now a part of passwd, by using the passwd –s option.

clear

clear terminal screen

The clear command clears your terminal screen if possible.

---

clear

DESCRIPTION

Clear looks in your user environment for your terminal type and then in /etc/termcap for the appropriate sequence of characters.

FILES

/etc/termcap	terminal capability data base

cmp

compare two files

The cmp command compares two files. By default, cmp makes no comment if the files are the same; if they differ, it announces the byte and line number at which the difference occurred. If one file is an initial subsequence of the other, that fact is noted.

---

cmp [-l] [-s] ***file file***

ARGUMENTS

-l displays the byte number (decimal) and the differing bytes (octal) for each difference.

-s displays nothing for differing files; returns codes only.

file specifies an input file. A hyphen specifies the standard input (the default).

DIAGNOSTICS

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

SEE ALSO

diff, comm

col

filter reverse line feeds

The col command performs the line overlays implied by reverse line feeds (ESC-7 in ASCII) and by forward and reverse half line feeds (ESC-9 and ESC-8). Col is particularly useful for filtering multi-column output made with the `.rt` command of nroff and output resulting from use of the tbl preprocessor. Col reads the standard input and writes the standard output.

---

col [-bfh]

ARGUMENTS

-b specifies that the output device in use is not capable of backspacing. In this case, if several characters are to appear in the same place, only the last one read is taken.

-f suppresses default treatment of half-line motions, as described below.

-h converts white space to tabs to shorten printing time.

DESCRIPTION

Although col accepts half line motions in its input, it normally does not emit them on output. Instead, text that would appear between lines is moved to the next lower full line boundary. This treatment can be suppressed by the -f (fine) option; in this case the output from col may contain forward half line feeds (ESC-9), but still never contains either kind of reverse line motion.

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

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

NOTES

- Col cannot back up more than 128 lines.
- The maximum number of characters, including backspaces, on a line is 800.

SEE ALSO

troff, tbl

colcrt

filter nroff output for crt previewing (unsupported)

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

---

colcrt [–] [–2] [*file...*]

ARGUMENTS

– suppresses all underlining. It is especially useful for previewing *allboxed* tables from tbl.

–2 causes all half-lines to be printed, effectively double spacing the output. Normally, a minimal space output format is used that suppresses empty lines. Colcrt never suppresses two consecutive empty lines, however. The –2 option is useful for sending output to the line printer when the output contains superscripts and subscripts that would otherwise be invisible.

EXAMPLE

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

NOTES

- Col cannot back up more than 102 lines.
- General overstriking is lost; as a special case, a vertical bar overstruck with a hyphen or an underline becomes a plus sign.
- Lines are trimmed to 132 characters.

SEE ALSO

nroff, troff, col, more, ul

colrm

remove columns from a file

The colrm command removes selected columns from a file. Input is taken from standard input. Output is sent to standard output. Column numbering starts with column one.

---

colrm [*startcol* [*endcol*]]

ARGUMENTS

startcol removes the columns of each line starting with the specified column.

endcol removes the columns of each line up to the specified column.

SEE ALSO

expand

comb

combine SCCS deltas

Comb generates a (Bourne) shell procedure that, when run, reconstructs the given SCCS files. The reconstructed files are, hopefully, smaller than the original files. The generated shell procedure is written on the standard output.

---

comb [-o] [-s] [-p*sid*] [-c*list*] *file...*

ARGUMENTS

The arguments can be specified in any order, but all keyletter arguments apply to all named SCCS files. Each keyletter argument is explained as though only one named file is to be processed, but the effects of any keyletter argument apply independently to each named file. If no keyletter arguments are specified, comb preserves only leaf deltas and the minimal number of ancestors needed to preserve the tree.

-o causes, for each get -e generated, the reconstructed file to be accessed at the release of the delta to be created, otherwise the reconstructed file would be accessed at the most recent ancestor. Use of the -o keyletter may decrease the size of the reconstructed SCCS file. It may also alter the shape of the delta tree of the original file.

-s causes comb to generate a shell procedure that, when run, produces a report giving, for each file: the file name, size (in blocks) after combining, original size (also in blocks), and percentage change computed by:

```
100 * (original - combined) / original
```

It is recommended that before any SCCS files are actually combined, use this option to determine exactly how much space is saved by the combining process.

-p*SID* causes the *S*CCS *ID*entification string (SID) of the oldest delta to be preserved. All older deltas are discarded in the reconstructed file.

-c*list* causes a *list* (see get for the syntax of a *list*) of deltas to be preserved. All other deltas are discarded.

files specifies the files to be processed. If a directory is named, *comb* behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored. If a name of "-" is given, the standard input is read; each line is taken to be the name of an SCCS file to be processed; non-SCCS files and unreadable files are silently ignored.

DIAGNOSTICS

- Use help for explanations.

NOTES

- Comb can rearrange the shape of the tree of deltas. It may not save any space; in fact, it is possible for the reconstructed file to actually be larger than the original.

FILES

s.COMB	name of reconstructed SCCS file
comb?????	temporary

SEE ALSO

sccs, admin, delta, get, help, prs
Concentrix Programming Tools Guide: Source Code Control System – System V Interface
Concentrix-2800 System Reference Manual: System Calls and C Library Subroutines: sccsfile

comm

compare files

The comm command displays a line-by-line comparison of two sorted files.

---

comm [**-123**] *file file*

ARGUMENTS

123 suppresses output of the corresponding column(s).

file specifies an input file. A hyphen specifies the standard input (the default).

DESCRIPTION

Comm produces a three-column output in the following format:

```
Column 1          Column 2          Column 3

lines only        lines only        lines in
in file1          in file2          both files
```

EXAMPLE

```
comm -12          displays only the lines common to the two files
comm -23          displays only lines unique to the first file
comm -123         displays nothing
```

NOTES

- Input files should be ordered in ASCII collating sequence.

SEE ALSO

cmp, diff, uniq

compact

compress files (old command, found in /usr/old, replaced by compress)

The compact command compresses the named files using an adaptive Huffman code.

---

compact [*name...*]

ARGUMENTS

name specifies an input file that is compacted and the resulting file placed in *name*.C before *name* is unlinked. If no file names are given, the standard input is compacted to the standard output.

DESCRIPTION

Compact operates as an on-line algorithm. Each time a byte is read, it is encoded immediately according to the current prefix code. This code is an optimal Huffman code for the set of frequencies seen so far. It is unnecessary to prepend a decoding tree to the compressed file since the encoder and the decoder start in the same state and stay synchronized. Furthermore, compact and uncompact can operate as filters.

The first two bytes of the compacted file code the fact that the file is compacted. This code is used to prohibit recompaction.

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

NOTES

- The last segment of the filename must contain fewer than thirteen characters to allow space for the appended .C.

FILES

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

SEE ALSO

uncompact, ccat
Variations on a Theme of Huffman, Gallager, Robert G.,
I.E.E.E. Transactions on Information Theory, vol. IT-24, no. 6,
November 1978, pp. 668 – 674.

compress

compress files

The compress command compresses the named files using adaptive Lempel–Ziv coding.

---

compress [–f] [–v] [–c] [–b*bits*] [*name...*]

ARGUMENTS

name specifies an input file that is compressed. The resulting file is placed in *name*.Z. If no file name is given, the standard input is compressed to the standard output.

–f forces the compression of *name*, even if the compression will not shrink the size of the file or the corresponding *name*.Z file already exists. Except when run in the background under /bin/sh or if –f is given, the user is prompted as to whether an existing *name*.Z file should be overwritten.

–v option causes compress to print the percentage reduction of each file.

–c ("cat") option makes *compress* write to the standard output; no files are changed. Be aware that compressed files may contain control characters, however. Writing such files to the screen may make the screen unreadable.

–b specifies the maximum size bit string that *compress* will use to encode common substrings in the file. The default is 16. See the description of the encoding algorithm below for more information.

DESCRIPTION

Compress reduces the size of the named files using adaptive Lempel–Ziv coding. Whenever possible, each file is replaced by one with the extension .Z, while keeping the same ownership modes, access and modification times. If no files are specified,the standard input is compressed to the standard output.

Compressed files can be restored to their original form using *uncompress* or *zcat*.

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

After the *bits* limit is attained, compress periodically checks the compression ratio. If it is increasing, compress continues to use the existing code dictionary. However, if the

compression ratio decreases, compress discards the table of substrings and rebuilds it from scratch. This allows the algorithm to adapt to the next block of the file.

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

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

If an error occurs, the exit status is1. If the last file was not compressed because it became larger, the status is 2. Otherwise the status is 0.

DIAGNOSTICS

- Usage: compress [-fvc] [-bmaxbits] [file ...]. Invalid options were specified on the command line.
- Missing maxbits. Maxbits must follow -b.
- *name* already has .Z suffix -- no change. The file is assumed to be already compressed. Rename the file and try again.
- *name*: filename too long to tack on.Z. The file cannot be compressed because its name is longer than 12 characters. Rename and try again. This message does not occur on BSD systems.
- *name* already exists; do you wish to overwrite (y or n)? Respond "y" if you want the output file to be replaced; "n" if not.
- Compression: xx.xx%. Percentage of the input saved by compression. (Relevant only for -v.)
- *name* not a regular file: unchanged. When the input file is not a regular file, (e.g. a directory), it is left unaltered.
- *name* -- has xx other links: unchanged. The input file has hard links; it is left unchanged. See ln(1) for more information about links.
- *name* -- file unchanged. No savings is achieved by compression. The input remains virgin.

NOTES

- Compressed files are compatible between machines with large memory. -b12 should be used for file transfer to architectures with a small (64KB or less) process data space, however, as exhibited by the DEC PDP series and the Intel 80286 line.

- Compress should be more flexible about the existence (or nonexistence) of the .Z suffix on compressed files.

- Compress does not work well with links. If a file has a hard link to it, compress will not compress the file, regardless of whether you use the file name or the link name.

  If a file has a symbolic link to it, compress will compress the file when you refer to it by the file name. After compression, however, the file is not accessible by its symbolic link name. *Compress* also will allow you to compress a file by using its symbolic link name, but *compress* then changes the symbolic link into an actual file.

FILES

*.Z	compressed file created by compress, removed by uncompress.

SEE ALSO

uncompress, zcat

Variations on a Theme of Huffman, Gallager, Robert G., I.E.E.E. Transactions on Information Theory, vol. IT-24, no. 6, November 1978, pp. 668 – 674.

continue

continue execution (C shell)

The continue command is intended for use in shell scripts. It continues execution of the nearest enclosing *foreach* or *while* loop. The rest of the commands on the current line are executed.

---

continue

NOTES

- The Bourne shell version of the continue command resumes the next iteration of the enclosing *for* or *while* loop. It accepts an optional numeric argument *n* that resumes at the *n*th enclosing loop.

SEE ALSO

foreach, while

continue

continue execution (Bourne shell)

The continue command is intended for use in shell scripts. It resumes the next iteration of the enclosing *for* or *while* loop. It accepts an optional numeric argument *n* that resumes at the *n*th enclosing loop.

---

continue [*n*]

ARGUMENTS

n Optional argument. If *n* is specified, resume at the *n*th enclosing loop.

SEE ALSO

for, while

cp

copy files, directories, and subtrees

The cp command duplicates the contents of one or more files at a specified destination. *Cp* refuses to copy a file onto itself.

---

cp [-ip] ***file1 file2***

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

ARGUMENTS

-i (inquire) prompts the user with the name of the file whenever the copy causes an old file to be overwritten. An answer of y causes cp to continue. Any other answer prevents it from overwriting the file.

-p causes cp to attempt to preserve the modification times and modes of the source files, ignoring the present umask.

-r (recursive) if any of the source files are directories, cp copies each subtree rooted at that name; in this case the destination must be a directory.

file1 is copied onto ***file2***.

file2 the destination of ***file1***.

file... are copied into ***directory*** with their original filenames.

directory the destination of ***file...***

DESCRIPTION

In the first form, ***file1*** is copied onto ***file2***. By default, the mode and owner of ***file2*** are preserved if it already existed; otherwise the mode of the source file modified by the current umask(2) is used. The -p option causes cp to attempt to preserve (duplicate) the modification times and modes of the source files, ignoring the present umask.

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

Cp refuses to copy a file onto itself.

SEE ALSO

cat, pr, mv

crypt

encode/decode files

The crypt command encrypts and decrypts files. It reads from the standard input and writes on the standard output.

---

crypt [*password*]

ARGUMENTS

password is a key that selects a particular transformation. If no *password* is given, crypt demands one and turns off echo while you type it in.

DESCRIPTION

Crypt encrypts and decrypts with the same *password*. For example:

```
crypt foobar <clear >cypher
crypt foobar <cypher | pr
```

displays the unencrypted file.

Files encrypted by crypt are compatible with those treated by the editor ed in encryption mode.

The security of encrypted files depends on three factors: the fundamental method must be hard to solve; direct search of the key space must be infeasible; "sneak paths" by which keys or cleartext can become visible must be minimized.

Crypt implements a one-rotor machine designed along the lines of the German Enigma, but with a 256-element rotor. Methods of attack on such machines are known, but not widely; moreover the amount of work required is likely to be large.

The transformation of a key into the internal settings of the machine is deliberately designed to be expensive, i.e. to take a substantial fraction of a second to compute. However, if keys are restricted to (say) three lower-case letters, encrypted files can be read by expending only a substantial fraction of five minutes of machine time.

Since the key is an argument to the crypt command, it is potentially visible to users executing ps or a derivative. To minimize this possibility, crypt takes care to destroy any record of the key immediately upon entry. No doubt the choice of keys and key security are the most vulnerable aspect of crypt.

NOTES

- There is no warranty of merchantability nor any warranty of fitness for a particular purpose nor any other warranty, either express or implied, as to the

accuracy of the enclosed materials or as to their suitability for any particular purpose. Accordingly, Alliant assumes no responsibility for their use by the recipient. Further, Alliant assumes no obligation to furnish any assistance of any kind whatsoever, or to furnish any additional information or documentation.

- This command is not included in international shipments of the software.

FILES

/dev/tty	for typed key

SEE ALSO

ed
Concentrix-2800 System Administation: makekey

csh

invoke C shell

The csh command invokes the C shell; a command language interpreter incorporating a history mechanism, filename completion function, job control facilities, and a C-like syntax.

---

csh [*option...*] [*script* [*arg...*]]

ARGUMENTS

- in argument 0 to the shell specifies a login shell.

-b forces a "break" from option processing, causing any further shell arguments to be treated as non-option arguments. The remaining arguments will not be interpreted as shell options. This may be used to pass options to a shell script without confusion or possible subterfuge. The shell will not run a set-user ID script without this option.

-c reads commands from the (single) following argument, which must be present (implies -f). Any remaining arguments are placed in *argv*.

-e exits if any invoked command terminates abnormally or yields a non-zero exit status.

-f neither search for nor execute commands from the file .cshrc in the invokers home directory, thus the shell starts faster (implied by -c).

-i prompt for top-level input, even if the source appears to not be a terminal. Shells are interactive by default if their inputs and outputs are terminals.

-n parse but do not execute commands (for use in syntactic checking of shell scripts).

-s take command input from the standard input.

-t read and execute a single line of input. A backslash can be used to escape the newline at the end of this line and continue onto another line.

-v set the *verbose* variable so that command input is echoed after history substitution.

-x set the *echo* variable so that commands are echoed immediately before execution.

-V set the *verbose* variable before .cshrc is executed.

-X set the *echo* variable before .cshrc is executed.

script specifies the name of a C shell script (cannot be used with –c, –i, –s, or –t). The shell opens this file, and saves its name for possible resubstitution by $0.

arg is one of a sequence of arguments passed to the C shell script (via the variable *argv*).

DESCRIPTION

The C shell begins operation by executing commands from the file `.cshrc` in the home directory of the invoker. A login C shell also executes commands from the file `.login` in the home directory. When a login shell terminates, it executes commands from the file .logout in the users home directory.

The C shell performs various input transformations in the order shown. Each transformation is described in its own section.

I/O REDIRECTION

< causes the standard input to come from a file.

<< causes the standard input to come from the text in the shell script. The word following the << (normally 'EOF') is used to mark the end of the text. The word that marks the end of the text must be on a line by itself, with no extra white space on either side. If the word is enclosed in single quotes, the shell does not perform variable substitution or command substitution on the data.

> redirects the standard output to a file. If the file does not exist, the shell creates a new one. If the file exists, it is overwritten, unless the shell variable *noclobber* is set.

>& redirects both the standard output and the diagnostic output to a file. If the file does not exist, the shell creates a new one. If the file exists, it is overwritten, unless *noclobber* is set.

>! is the same as > except that the shell variable *noclobber* is ignored.

>&! is the same as >& except that the shell variable *noclobber* is ignored.

>> redirects the standard output to the end of a file. If *noclobber* is set, the file must already exist.

>>& redirects both the standard output and the diagnostic output to the end of a file. If the file does not exist, the shell creates a new one. If the file exists, it is overwritten, unless *noclobber* is set.

>>! is the same as >> except that the shell variable *noclobber* is ignored.

>>&! is the same as >>& except that the shell variable *noclobber* is ignored.

| redirects the standard output of a program to the standard input of the next program.

|& redirects both the standard and diagnostic output of one program to the input of another.

`||` redirects the standard output of a program to the standard input of the next program only if the first program fails.

`&&` redirects the standard output of a program to the standard input of the next program only if the first program succeeds.

HISTORY SUBSTITUTION

`!`*n* where *n* is an integer, refers to the *n*th event in the history list.

`!-`*n* where *n* is an integer, refers to the *n*th event counting backward from the current event in the history list.

`!!` refers to the previous event (equivalent to `!-1`).

`!`*c* where *c* is a string, refers to the most recent event that begin with the specified string.

`!?`*c*`?` where *c* is a string, refers to the most recent event that contained the specified string.

`^`*o*`^`*n* refers to the previous event, replacing the first occurrence of the old (*o*) string with the new (*n*) string.

Word Selectors

n specifies the *n*'th word from the left, numbered from 0.

`^` specifies the second word (usually the first argument).

`$` specifies the last word.

`%` specifies the word matched by the immediately preceding substring search.

x`-`*y* specifies a range of words, *x* to *y*. You can use designators in a range.

`-`*y* is equivalent to 0-*y*.

`*` is equivalent to `^-$`, or nothing if there was only one word in the event.

x`*` is equivalent to *x*`-$`.

x`-` is similar to *x*`*` but omits the last word (`$`).

Colon Modifiers

h removes a trailing pathname component, leaving the head.

r removes a trailing extension (*.xxx*) component, leaving the root name.

e removes all but the extension (*.xxx*) component.

s/*o*/*n*/ replaces the first occurrence of the old (*o*) string with the new (*n*) string.

t (tail) removes all but the last component from a pathname.

& repeats the previous substitution.

g combined with another modifer (such as s), applies the change to each word of the event.

p prints the event, rather than executing it.

q quotes the substituted words, preventing further substitutions.

x is similar to q, but breaks into words at blanks, tabs and newlines.

FILENAME COMPLETION

When the file name completion feature is enabled by setting the shell variable filec (see set), *csh* will interactively complete file names and user names from unique prefixes, when they are input from the terminal followed by the escape character (the escape key, or CTRL[). For example, if the current directory looks like

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

and the input is

```
% vi ch<escape>
```

csh will complete the prefix "ch" to the only matching file name "chaosnet", changing the input line to

```
% vi chaosnet
```

However, given

```
% vi D<escape>
```

csh will only expand the input to

```
% vi DSC.
```

and will sound the terminal bell to indicate that the expansion is incomplete, since there are two file names matching the prefix "D".

If a partial file name is followed by the end-of-file character (usually CTRL D), then, instead of completing the name, *csh* will list all file names matching the prefix. For example, the input

```
% vi DCTRL D
```

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

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

while the input line remains unchanged.

The same system of escape and end–of–file can also be used to expand partial user names, if the word to be completed (or listed) begins with the character "~". For example, typing

```
cd ~roCTRLD
```

may produce the expansion

```
cd ~root
```

The use of the terminal bell to signal errors or multiple matches can be inhibited by setting the variable **nobeep**.

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

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

then typing

```
% vi x<escape>
```

would result in the completion to

```
% vi xmpl.c
```

ignoring the files "xmpl.o" and "xmpl.out". However, if the only completion possible requires not ignoring these suffixes, then they are not ignored. In addition, **fignore** does not affect the listing of file names by CTRL D. All files are listed regardless of their suffixes.

QUOTATION

\ prevents the next character from having a special meaning. A newline preceded by a backslash is equivalent to a blank.

' ' (close single quotes) prevent all substitutions except history substitutions.

" " (double quotes) prevent alias substitutions and filename substitutions.

ALIAS SUBSTITUTION

After a command line is scanned, it is parsed into distinct commands and the first word of each command, left-to-right, is checked to see if it has an alias. If it does, the text that is the alias for that command is reread (with the history mechanism available) as though that command were the previous input line. The resulting words replace the command and argument list. If no reference is made to the history list, the argument list is left unchanged.

VARIABLE SUBSTITUTION

Except as noted, it is an error to reference a variable that is not set.

$name
${name}
: invokes variable substitution (is replaced by the value of the variable *name*); the braces insulate *name* from any following characters that would otherwise be part of it. If *name* is not a shell variable, but is set in the environment, that value is returned (but modifiers and other forms are not available).

$name[index]
${name[index]}
: specifies some of the elements of the value of *name* (counting from one); *index* is subjected to variable substitution and can consist of a single number or a range (two numbers separated by a hyphen). If the first number of a range is omitted, the default is `1`. If the last member of a range is omitted, the defaults is *$#name*. Asterisk (*) selects all words. A range can be empty if the second argument is omitted or in range.

$?name
${?name}
: returns 1 if the variable *name* exists (has been assigned a value); 0 otherwise(cannot be modified).

$#name
${#name}
: returns the number of elements in the variable *name*.

$number
${number}
: is equivalent to `$argv[`*number*`]`.

$*
: is equivalent to `$argv[*]`.

$0
: returns the name of the file from which command input is being read. An error occurs if the name is not known.

$?0
: substitutes `1` if the current input filename is known, `0` if it is not (cannot be modified).

$$
: returns the (decimal) process number of the (parent) shell (cannot be modified).

$<
: returns a line from the standard input, with no further interpretation thereafter (cannot be modified). It can be used to read from the keyboard in a shell script.

Colon Modifiers

If braces appear in the command form, the modifiers must appear within the braces. Only one colon modifier (other than g) is allowed on each variable substitution.

h
: (head) removes a trailing pathname component, leaving the head.

r
: (root) removes a trailing extension component, leaving the root name.

e
: (extension) removes all but the extension component.

t (tail) removes all leading pathname components, leaving the tail.

g (global) applies the change globally, prefixing another modifier.

q (quote) quotes the substituted words, preventing further substitutions.

x is similar to q, but breaks into words at blanks, tabs and newlines.

COMMAND SUBSTITUTION

'*c*' (open single quotes) causes a command *c* to be executed by the C shell, then replaced by its own output.

FILENAME SUBSTITUTION

***** matches any string of characters, including the empty string.

? matches any single character in a filename.

[] contains single-character numbers and letters that specify a set. A hyphen within a set indicates a range.

{} contains a set of strings, separated by commas, that are consecutively substituted into the preceding and subsequent characters and the results expanded left to right. For example: `a{b,c,d}e` produces `abe ace ade`.

~ followed immediately by a user name expands to the *home* directory of the specified user. The default is your own home directory.

OTHER METACHARACTERS

; delimits multiple commands on a single input line.

() causes multiple commands to execute in a separate subshell.

except when the shell's input is a terminal, introduces a comment that continues to the end of the input line.

OPERATORS

A number of the built-in commands take expressions in which the operators are similar to those of C. The following operators (shown in descending order of precedence) operate on numbers except where indicated:

( )	grouping
!	negation
%	remainder
/	division
*	multiplication
-	subtraction
+	addition

>>	shift right
<<	shift left
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
!-	not equal to (pattern string)
=-	equal to (pattern string)
!=	not equal to (string)
==	equal to (string)
&	bitwise and
^	bitwise exclusive or
|	bitwise inclusive or
&&	logical and
||	logical or

The following groups have the same level of precedence: (== != =- !-), (<= >= < >), (<< >>), (+ -), (* / %).

The operators =- and !- are like != and == except that the right hand side is a *pattern* (containing filename substitution characters) against which the left hand operand is matched. This reduces the need for use of the *switch* command in shell scripts when all that is really needed is pattern matching.

NUMERIC OPERANDS

Strings that begin with 0 are considered octal numbers. Null or missing arguments are considered 0. The result of all expressions are strings that represent decimal numbers. It is important to note that no two components of an expression can appear in the same word; except when adjacent to components of expressions that are syntactically significant to the parser (ampersand, vertical bar, angle brackets, and parentheses) they should be surrounded by spaces.

FILE OPERANDS

An operand of the following form provides true (1) or false (0) information about the specified file:

-c filename

where *c* is one or more of the following characters:

d	directory
e	existence
f	plain file
o	ownership
r	read access
w	write access
x	execute access
z	zero size

COMMAND EXPRESSIONS

Use an operand of the following form tests whether a command terminates normally:

{ *command* }

The expression returns `1` (true) if the command succeeds (exits normally with exit status zero), or `0` (false) if the command fails (terminates abnormally or with exit status non-zero.) For more detailed information about the execution status of a command, examine the shell variable `status`.

NOTES

- A *cmdfile* which has execute permission but is not a binary executable and is not a shell script directly executable by Concentrix-2800 is further examined by the shell which is attempting to execute the command. If the first character is a #(pound sign) then *cmdfile* is deemed to be a file of csh commands. If the first character is NOT a #, then *cmdfile* is deemed to be a file of sh commands. If necessary, one shell will exec the other to handle the commands.

  The csh somewhat complicates this by using the string associated with the alias "shell" as the interpreter to use. Failing that, it examines the first character and uses /bin/sh if it is not #. Otherwise (if it is #), csh examines the shell variable "shell" and uses the interpreter associated with it. Finally, failing that, it falls back to using /bin/csh.

  Both shells exec the interpreter which is finally chosen with *cmdfile* as the argument.

- Symbolic links fool the shell. In particular, *dirs* and `cd ..` don't work properly once you've crossed through a symbolic link.

FILES

`~/.cshrc`	Read at beginning of execution by each shell.
`~/.login`	Read by login shell, after `.cshrc` at login.
`~/.logout`	Read by login shell, at logout.
`/bin/sh`	Standard shell, for scripts not starting with a sharp sign.
`/tmp/sh*`	Temporary file for <<.
`/etc/passwd`	Source of home directories for ~name.

BUILTIN COMMANDS

csh Builtin commands are listed in this section. A man page is supplied for each one individually.

alias	bind name to string
alloc	show amount of dynamic core in use
bg	execute a job in the background

break	resume execution after foreach or while
breaksw	break from switch command
case	denote label in switch command
cd	change working directory
chdir	change working directory
continue	continue execution
default	default case in switch command
dirs	display directory stack
echo	echo arguments
else	alternate branch in if–then–else
end	terminate control structure
endif	terminate if–then–else
endsw	terminate switch
eval	evaluate arguments and execute resulting command
exec	execute command in place of current shell
exit	exit from shell
fg	execute job in foreground
foreach	iterated loop construct
glob	echo arguments delimited by nulls
goto	unconditional branch control structure
hashstat	print hash table statistics
history	display most recent commands
if	simple two-way branch control structure
if–then–else	multiple two-way branch control structure
jobs	list active jobs
kill	terminate job or process
limit	limit resource consumption

login	create new user process
logout	terminate login shell
nice	set priority for command
nohup	ignore hangup signals
notify	display job status changes
onintr	enable interrupt handling
popd	pop directory stack
pushd	push directory onto stack
rehash	recompute path table
repeat	execute command repetitively
set	set local variable
setenv	set environment variable
shift	shift members of variable
source	execute commands from file
stop	stop background job
suspend	stop current shell
switch	multi-way branch control structure
time	time a command
umask	set file creation mask
unalias	remove alias
unlimit	remove resource limitation
unset	remove local variable
unsetenv	remove environmental variable
wait	wait for background jobs
while	test and loop control stucture

SEE ALSO

sh
Concentrix-2800 System Reference Manual: System Calls and C Library Subroutines: access, execve, execvec, fork, killpg, pipe, sigvec, umask, setrlimit, wait
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: a.out, tty
Concentrix System User's Guide: Using the C Shell, Programming the C Shell

ctags

create tags file

The ctags command makes a tags file for ex from the specified C, Pascal, Fortran, YACC, lex and lisp sources. A tags file gives the locations of specified objects (in this case functions and typedefs) in a group of files.

---

ctags [-BFatuwvx] [-f *filename*] *name...*

ARGUMENTS

-B uses backward searching patterns (?...?).

-F uses forward searching patterns (/.../) (default).

-a appends to tags file.

-t creates tags for typedefs.

-u causes the specified files to be *updated* in tags, that is, all references to them are deleted, and the new values are appended to the file. (Beware: this option is implemented in a way that is rather slow; it is usually faster to simply rebuild the tags file.)

-w suppresses warning diagnostics.

-v produces an index of the form expected by *vgrind* on the standard output. The listing contains the function name, file name, and page number (assuming 64 line pages). Since the output is sorted into lexicographic order, it may be desired to run the output through sort -f.

-x produces a list of object names on the standard output containing the line number and file name on which each is defined, as well as the text of that line. The list is a simple index that can be printed out as an off-line readable function index.

-f *filename* puts the tag descriptions in *filename*. If this option is not used, tag descriptions are put in a file named tags.

DESCRIPTION

Each line of the tags file contains the object name, the file in which it is defined, and an address specification for the object definition. Functions are searched with a pattern, typedefs with a line number. Specifiers are given in separate fields on the line, separated by blanks or tabs. Using the tags file, ex can quickly find these objects definitions.

Files whose names end in .c or .h are assumed to be C source files and are searched for C routine and macro definitions. Files whose names end in .y are assumed to be YACC

sources. Files whose names end in .l are assumed to be lisp sources if their first non-blank character is ; (semicolon), ((left parenthesis), or [(left brace). If a .l file does not begin with one of these characters, it is assumed to be lex source.

All other files are first examined to see if they contain any Pascal or Fortran routine definitions; if not, they are processed again looking for C definitions.

The tag *main* is treated specially in C programs. The tag formed is created by prepending *M* to the name of the file, with a trailing .c removed, if any, and leading pathname components also removed. This makes use of ctags practical in directories with more than one program.

NOTES

- Recognition of functions, subroutines and procedures for FORTRAN and Pascal is done in a very simpleminded way. No attempt is made to deal with block structure; if you have two Pascal procedures in different blocks with the same name you lose.
- The method of deciding whether to look for C or Pascal and FORTRAN functions is a hack.
- Ctags does not know about #ifdefs.
- Ctags should know about Pascal types. It relies on the input being well formed to detect typedefs. Use of -tx shows only the last line of typedefs.

EXAMPLE

```
ctags -v files | sort -f > index
vgrind -x index
```

FILES

tags	output tags file

SEE ALSO

cref, ex, vi
GNU Emacs Manual

ctime

outputs the time in hexadecimal

The ctime command displays the date and time in hexadecimal format.

---

ctime

NOTES

- You can get the same information, in a more readable form, from the date command.

date

display and set the date

The date command displays or sets the system date and time.

---

date [-n] [-u] [-d *dst*] [-t *minutes_west*][*yymmddhhmm*[*.ss*]]

ARGUMENTS

If no arguments are given, the current date and time are displayed. If a date is specified, the current date is set. The year, month and day can be omitted, the current values being the defaults.

-n do not set the time throughout a local area network.

-u displays or sets the date in GMT (universal) time.

-d *dst* Time zone uses US daylight savings time. Enter 0 for *dst* if the time zone uses standard time only, and 1 if US daylight savings time is used.

-t *minutes_west* time is minutes west of GMT. This number can be negative.

yy is a two-digit number that specifies the current year.

mm is a two-digit number that specifies the current month.

dd is a two-digit number that specifies the current day.

hh is a two-digit number that specifies the current hour (24 hour system).

mm is a two-digit number that specifies the current minute.

.ss is a two-digit number (preceded by a period) that specifies the current second.

DESCRIPTION

Only the superuser can set the date. See the *Concentrix-2800 System Administration Guide*, Chapter 1, "Date and Time" for complete information on setting the date and time.

The system operates in GMT. Date takes care of the conversion to and from local standard and daylight time.

If timed is running to synchronize the clocks on a local area network, date will set the time globally across the systems. The -n option disables network-wide time setting.

EXAMPLE

```
% date
Tue Oct 30 12:19:39 EST 1990
# date 9006131627          sets the date to June 13 1990, 4:27 p.m.
# date -d 1 -t 480 9006131627    sets the date to June 13 1990, 4:27
                                  p.m. on the US West Coast
```

DIAGNOSTICS

- `Failed to set date: Not owner` – if you try to change the date but are not the superuser.
- Exit status is 0 on success, 1 on complete failure to set the date, 2 on successfully setting the date locally but not globally.
- `Network time being set` — date is waiting while timed is setting the time on other systems.
- `Communication error with timed` — there is a communication problem between date and timed.

FILES

/usr/adm/wtmp	to record time setting
/usr/adm/messages	to record the name of the people who set the time
/etc/zoneinfo	to set geographical time zone

SEE ALSO

Concentrix-2800 System Reference: System Calls and C Library Subroutines: gettimeofday
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: utmp
Concentrix-2800 System Administration: timed, timezone
ISP: The Time Synchronization Protocol for UNIX 4.3BSD, R. Gusella and S. Zatti.

dbx

source-level debugger

The dbx command invokes a tool for source level debugging and execution of programs. Currently, fortran, pc and cc produce the appropriate source information. The machine-level facilities of dbx can be used on any program.

---

dbx [-r] [-i] [-s *file*] [-sr *file*] [-I *dir*] [-c *dir*] [*objfile* [*coredump*]]

ARGUMENTS

-r executes *objfile* immediately. By default, dbx just prompts and waits for a command. If *objfile* terminates successfully dbx exits. Otherwise the reason for termination is reported and the user is offered the option of entering the debugger or letting the program fault. Dbx reads from `/dev/tty` when -r is specified and the standard input is not a terminal.

-i forces dbx to act as though the standard input is a terminal.

-s *file* at startup, reads commands from *file* instead of .dbxinit.

-sr at startup, reads commands from *file* instead of .dbxinit, then deletes *file*.

-I *dir* adds *dir* to the list of directories that are searched when looking for a source file. Normally dbx looks for source files in the current directory and in the directory containing *objfile*. The directory search path can also be set with the use command.

-c *file* executes the dbx commands in *file* before reading from standard input.

objfile is an object file containing a symbol table that includes the name of all the source files translated by the compiler to create it (usually produced by the -g command line option). By default, dbx looks for a file named `a.out` in the current directory. The files are available for perusal while using dbx.

coredump specifies a core dump file. If a file named `core` exists in the current directory or a core dump file is specified, dbx can be used to examine the state of the program when it faulted.

DESCRIPTION

Dbx initially executes the commands in one of the following files: the file specified by the **-s** or **-sr** option, the file named .dbxinit in the current directory, or the file named .dbxinit in the user's home directory.

EXAMPLE

The following example invokes dbx for an object file named balance, ignoring any core file that may be present. When the (dbx) prompt appears, dbx is ready to accept commands. To exit dbx, use the quit command.

```
% dbx balance -
dbx version 1.0.11
initializing...
reading symbolic information...
read 376 symbols
(dbx)
```

NOTES

- Non-local gotos can cause some trace/stops to be missed.
- Dbx suffers from the multiple include malady. If you have a program consisting of a number of object files and each is built from source files that include header files, the symbolic information for the header files is replicated in each object file. Since about one debugger start-up is done for each link, having the linker (ld) reorganize the symbol information does not save much time, though it reduces some of the disk space used. The problem is an artifact of the unrestricted semantics of #include's in C; for example, an include file can contain static declarations that are separate entities for each file in which they are included.

FILES

a.out	object file
.dbxinit	initial commands
core	core dump file

SEE ALSO

adb, cc
FX/Fortran-2800 Programmer's Handbook, Appendix G, "DBX Source-Level Debugger"
FX/C-2800 C Programmers's Handbook, Appendix G, "DBX Source-Level Debugger"

dc

desk calculator

The dc command invokes an arbitrary precision arithmetic package. Ordinarily it operates on decimal integers, but one can specify an input base, output base, and a number of fractional digits to be maintained.

---

dc [*file*]

ARGUMENTS

file specifies an input file. Input is taken from that file until its end, then from the standard input.

DESCRIPTION

The overall structure of dc is a stacking (reverse Polish) calculator. The following constructions are recognized:

number
: pushes the value of the number onto the stack. A number is an unbroken string of the digits 0–9. It can be preceded by an underscore _ to input a negative number. Numbers can contain decimal points.

+ – / * % ^
: adds (+), subtracts (–), multiplies (*), divides (/), remainders (%), or exponentiates (^) the top two values on the stack. The two entries are popped off the stack; the result is pushed on the stack in their place. Any fractional part of an exponent is ignored.

s*x*
: pops the top of the stack and stores the result in a register named *x*, where *x* can be any character. If the s is capitalized, *x* is treated as a stack and the value is pushed on it.

l*x*
: pushes the value in register *x* on the stack. The register *x* is not altered. All registers start with zero value. If the l is capitalized, register *x* is treated as a stack and its top value is popped onto the main stack.

d
: duplicates the top value on the stack.

p
: displays the top value on the stack. The top value remains unchanged. P interprets the top of the stack as an ASCII string, removes it, and displays it.

f
: displays all values on the stack and in registers.

q
: exits the program. If executing a string, the recursion level is popped by two. If q is capitalized, the top value on the stack is popped and the string execution level is popped by that value.

x treats the top element of the stack as a character string and executes it as a string of dc commands.

X replaces the number on the top of the stack with its scale factor.

[...] puts the bracketed ASCII string onto the top of the stack.

<*x* >*x* =*x*
The top two elements of the stack are popped and compared. Register *x* is executed if they obey the stated relation.

v replaces the top element on the stack by its square root. Any existing fractional part of the argument is taken into account, but otherwise the scale factor is ignored.

! interprets the rest of the line as a shell command.

c pops all values on the stack.

i pops the top value on the stack and uses it as the number radix for further input.

I pushes the input base on the top of the stack.

o pops the top value on the stack and uses it as the number radix for further output.

O pushes the output base on the top of the stack.

k pops the top of the stack and uses it as a non-negative scale factor: the appropriate number of places are displayed on output, and maintained during multiplication, division, and exponentiation. The interaction of scale factor, input base, and output base will be reasonable if all are changed together.

z pushes the stack level onto the stack.

Z replaces the number on the top of the stack with its length.

? takes and executes a line of input from the input source (usually the terminal).

; : are used by *bc* for array operations.

DIAGNOSTICS

- `x is unimplemented` – where x is an octal number.
- `stack empty` – for not enough elements on the stack to do what was asked.
- `Out of space` – when the free list is exhausted (too many digits).
- `Out of headers` – for too many numbers being kept around.
- `Out of pushdown` – for too many items on the stack.
- `Nesting Depth` – for too many levels of nested execution.

SEE ALSO

bc
Concentrix Programming Tools Guide: DC – An Interactive Desk Calculator
Concentrix Programming Tools Guide: BC – Desk-Calculator Language

dd

convert and copy a file

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

---

dd [*option=value*]...

ARGUMENTS

option	*values*
if=	input file name; standard input is default.
of=	output file name; standard output is default.
ibs=*n*	input block size *n* bytes (default 512).
obs=*n*	output block size (default 512).
bs=*n*	set both input and output block size,superseding *ibs* and *obs*; also, if no conversion is specified, it is particularly efficient since no copy need be done.
cbs=*n*	conversion buffer size.
skip=*n*	skip *n* input records before starting copy.
files=*n*	copy *n* input files before terminating (makes sense only where input is a magtape or similar device).
seek=*n*	seek *n* records from beginning of output file before copying (see note).
iseek=*n*	seek *n* records from beginning of input file before copying.
oseek=*n*	seek *n* records from beginning of output file before copying.
dseek=*n*	seek *n* records from beginning of input and output files before copying.
count=*n*	copy only *n* input records.
conv=ascii	convert EBCDIC to ASCII.
conv=ebcdic	convert ASCII to EBCDIC.
conv=ibm	slightly different map of ASCII to EBCDIC.

conv=block convert variable length records to fixed length.

conv=unblock convert fixed length records to variable length.

conv=lcase map alphabetics to lower case.

conv=ucase map alphabetics to upper case.

conv=swab swap bytes in every word.

conv=swal swap bytes in every longword.

conv=swaq swap bytes in every quadword.

conv=noerror do not stop processing on an error.

conv=sync pad every input record to *ibs*.

conv=... , ... several comma-separated conversions.

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

Cbs is used only if *ascii*, *unblock*, *ebcdic*, *ibm*, or *block* conversion is specified. In the first two cases, *cbs* characters are placed into the conversion buffer, any specified character mapping is done, trailing blanks trimmed and newline added before sending the line to the output. In the latter three cases, characters are read into the conversion buffer, and blanks added to make up an output record of size *cbs*.

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

EXAMPLE

To read an EBCDIC tape blocked ten 80-byte EBCDIC card images per record into the ASCII file *x*:

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

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

DIAGNOSTICS

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

NOTES

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

- Some combinations of arguments to conv= are permitted. However, the block or unblock options cannot be combined with ascii, ebcdic or ibm. Invalid combinations silently ignore all but the last mutually exclusive keyword.

- One must specify "conv=noerror,sync" when copying raw disks with bad sectors to ensure that dd stays synchronized.

- The seek argument seeks records one block at a time and can take a long time for large *n* values. The iseek, oseek, and dseek arguments do one seek only.

- Dd executes the skip argument before any seek arguments, if both are present.

SEE ALSO

cp, tr

default

default case in switch command (C shell)

The default command is intended for use in shell scripts. It labels the default case in a switch command.

---

default

NOTES

- The default command should come after all case labels.

SEE ALSO

switch

delta

make a delta (change) to an SCCS file

The delta command permanently introduces into the named SCCS file, changes that were made to the file retrieved by get (called the *g-file*, or generated file). Delta can issue prompts on the standard output depending upon certain keyletters specified and flags (see admin) that may be present in the SCCS file (see -m and -y keyletters below).

---

delta [-r*SID*] [-s] [-n] [-g*list*] [-m[*mrlist*]] [-y[*comment*]] [-p] *file...*

ARGUMENTS

Keyletter arguments apply independently to each named file.

-r*SID* uniquely identifies which delta is to be made to the SCCS file. The use of this keyletter is necessary only if two or more outstanding gets for editing (get -e) on the same SCCS file were done by the same person (login name). The SID value specified with the -r keyletter can be either the SID specified on the get command line or the SID to be made as reported by the get command (see get). A diagnostic results if the specified SID is ambiguous, or, if necessary and omitted on the command line.

-s suppresses the issue, on the standard output, of the created delta's SID, as well as the number of lines inserted, deleted and unchanged in the SCCS file.

-n specifies retention of the edited *g-file* (normally removed at completion of delta processing).

-g*list* specifies a *list* (see get for the definition of *list*) of deltas that are to be *ignored* when the file is accessed at the change level (SID) created by this delta.

-m[*mrlist*]
if the SCCS file has the v flag set (see admin), a Modification Request (MR) number *must* be supplied as the reason for creating the new delta.

If -m is not used and the standard input is a terminal, the prompt MRs? is issued on the standard output before the standard input is read; if the standard input is not a terminal, no prompt is issued. The MRs? prompt always precedes the comments? prompt (see -y keyletter).

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

Note that if the v flag has a value (see admin), it is taken to be the name of a program (or shell procedure) that validates the correctness of the MR numbers. If a non-zero exit status is returned from MR number validation program, delta terminates (it is assumed that the MR numbers were not all valid).

-y[*comment*]
specifies arbitrary text describing the reason for making the delta. A null string is considered a valid *comment*.

If -y is not specified and the standard input is a terminal, the prompt comments? is issued on the standard output before the standard input is read; if the standard input is not a terminal, no prompt is issued. An unescaped new-line character terminates the comment text.

-p causes delta to print (on the standard output) the SCCS file differences before and after the delta is applied in a diff format.

files specifies files to be processed. If a directory is named, delta behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored. If a hyphen is given, the standard input is read (see WARNINGS); each line of the standard input is taken to be the name of an SCCS file to be processed.

WARNINGS

- Lines beginning with an SOH ASCII character (binary 001) cannot be placed in the SCCS file unless the SOH is escaped. This character has special meaning to SCCS (see the file format sccsfile) and causes an error.
- A *get* of many SCCS files, followed by a delta of those files, should be avoided when the *get* generates a large amount of data. Instead, multiple get/delta sequences should be used.
- If the standard input (-) is specified on the delta command line, the -m (if necessary) and -y keyletters *must* also be present. Omission of these keyletters causes an error to occur.
- Comments are limited to text strings of at most 512 characters.

DIAGNOSTICS

- Use help for explanations.

FILES

All files of the form ?-file are explained in *Concentrix Programming Tools Guide: Source Code Control System - System V Interface*. The naming convention for these files is also described there.

g-file Existed before the execution of delta; removed after completion of delta.

p-file Existed before the execution of delta; can exist after completion of delta.

q-file Created during the execution of delta; removed after completion of delta.

x-file Created during the execution of delta; renamed to SCCS file after completion of delta.

z-file Created during the execution of delta; removed during the execution of delta.

d-file Created during the execution of delta; removed after completion of delta.

`/usr/bin/bdiff`
Program to compute differences between the "gotten" file and the *g-file*.

SEE ALSO

admin, bdiff, cdc, get, help, prs, rmdel
Concentrix Programming Tools Guide: Source Code Control System – System V Interface
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: sccsfile

deroff

remove nroff, troff, tbl, eqn constructs (unsupported)

The deroff command removes all nroff and troff command lines, backslash constructions, macro definitions, *eqn* constructs (between .EQ and .EN lines or between delimiters), and table descriptions from a file and writes the remainder on the standard output. Deroff follows chains of included files (.so and .nx commands); if a file has already been included, a .so is ignored and a .nx terminates execution.

---

deroff [-w] *file...*

ARGUMENTS

-w outputs a word list, one "word" (string of letters, digits, and apostrophes, beginning with a letter; apostrophes are removed) per line, and all other characters ignored. Otherwise, the output follows the original, with the deletions mentioned above.

file specifies an input file. Deroff reads each file in sequence. If no input file is given, deroff reads from the standard input file.

NOTES

- Deroff is not a complete troff interpreter, so it can be confused by subtle constructs. Most errors result in too much rather than too little output.

SEE ALSO

troff, eqn, tbl
Document Preparation Guide

df

report free disk space on file systems

The df command displays the amount of disk space occupied by currently mounted file systems, the amount of used and available space, and how much of the file system's total capacity has been used.

---

df [-i] [-n] [*filesystem* ...] [*file* ...]

ARGUMENTS

-i reports the number of inodes that are used and free.

-n ignore NFS type file systems.

filesystem specifies a filesystem.

file specifies a file.

If no *filesystem* or *file* is specified, the free space on all of the normally mounted file systems is displayed.

DESCRIPTION

The df command displays the amount of disk space occupied by currently mounted file systems, the amount of used and available space, and how much of the file system's total capacity has been used. Used without arguments, df reports on all mounted file systems, producing something like:

```
% df

Filesystem kbytes used avail capacity Mounted on

/dev/ip0a   7445  4714  1986  70%    /
/dev/ip0g  42277 35291 2758  93%    /usr
```

Note that used+avail is less than the amount of space in the file system (kbytes); this is because the system reserves a fraction of the space in the file system to allow its file system allocation routines to work well. The amount reserved is typically about 10%; this may be adjusted using tunefs. When all the space on a file system except for this reserve is in use, only the super-user can allocate new files and data blocks to existing files. When a file system is overallocated in this way, df may report that the file system is more than 100% utilized.

If arguments to df are disk partitions (for example, /dev/ip0a) or UNIX path names, df produces a report on the file system containing the named file. Thus "df ." shows the amount of space on the file system containing the current directory.

FILES

/etc/mtab	list of normally mounted filesystems

SEE ALSO

du
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: mstab
Concentrix-2800 System Administration: tunefs

diction

find wordy sentences in document (unsupported)

The diction command finds all sentences in a document that contain phrases from a database of bad or wordy diction. Each phrase is indicated with square brackets.

---

diction [-ml] [-mm] [-n] [-f *pfile*] *file...*

ARGUMENTS

-ml	skips lists and should be used if the document contains many lists of non-sentences.
-mm	overrides the default macro package -ms.
-n	suppresses the default pattern file.
-f *pfile*	specifies a pattern file to be used in addition to the default file.
file	specifies a document to check.

DESCRIPTION

Because diction runs deroff before looking at the text, formatting header files should be included as part of the input.

NOTES

- Use of non-standard formatting macros can cause incorrect sentence breaks. In particular, diction doesn't understand -me.

SEE ALSO

deroff, explain
Document Preparation Guide: The Style and Diction Programs

diff

compare files and directories

The diff command compares files and reports which lines must be changed to bring them into agreement. Except in rare circumstances, diff finds a smallest sufficient set of file differences.

---

diff [-l] [-r] [-s] [-S*name*] [-cefhn] [-biwt] ***dir1 dir2***

diff [-cefhn] [-biwt] ***file1 file2***

diff [-D*string*] [-biw] ***file1 file2***

ARGUMENTS

-l specifies long output format; each text file *diff* is piped through *pr* to paginate it, other differences are remembered and summarized after all text file differences are reported.

-r causes application of diff recursively to common subdirectories encountered.

-s causes diff to report files that are the same, which are otherwise not mentioned.

-S*name* starts a directory diff in the middle beginning with file *name*.

dir1, dir2 if both arguments are directories, diff sorts the contents of the directories by name, and then runs the regular file comparison algorithm on text files that are different. Binary files that differ, common subdirectories, and files that appear in only one directory are listed.

-c produces a comparison with lines of context. The default is to present three lines of context and can be changed, e.g. to 10, by -c10. With -c the output format is modified slightly: the output beginning with identification of the files involved and their creation dates and then each change is separated by a line with a dozen askterisks. The lines removed from *file1* are marked with a hyphen; those added to *file2* are marked with a plus sign. Lines that are changed from one file to the other are marked in both files with an exclamation point.

-e produces a script of *a*, *c* and *d* commands for the editor ed, that recreates *file2* from *file1*. In connection with -e, the following shell program can help maintain multiple versions of a file. Only an ancestral file ($1) and a chain of version-to-version *ed* scripts ($2,$3,...) made by diff need be on hand. A 'latest version' appears on the standard output.

```
(shift; cat $*; echo '1,$p') | ed - $1
```

Extra commands are added to the output when comparing directories with -e, so that the result is a sh script for converting text files that are common to the two directories from their state in *dir1* to their state in *dir2*.

-f produces a script similar to that of -e, not useful with *ed*, and in the opposite order.

-h does a fast, half-hearted job. It works only when changed stretches are short and well separated, but does work on files of unlimited length.

-n produces a script similar to that of -e, but in the opposite order and with a count of the changed lines on each insert and delete command. This is the form used by rcsdiff.

-D*string* creates a merged version of *file1* and *file2* on the standard output, with C preprocessor controls included so that a compilation of the result without defining *string* is equivalent to compiling *file1*, while defining *string* yields *file2*.

-b ignores trailing blanks (spaces and tabs), and causes other strings of blanks to compare equal.

-w is similar to -b but causes whitespace (spaces and tabs) to be totally ignored. For example, "if (a == b)" will be seen as equal to "if(a==b)".

-i ignores the case of letters. For example, "A" will be considered equal to "a".

-t expands tabs in output lines. Normal or -c output adds characters to the front of each output line. These extra characters may screw up the indentation of the original source lines and make the output listing difficult to read. This option preserves the original sources indentation.

file1, file2 if neither file is a directory, either can be a hyphen, which specifies the standard input. If *file1* is a directory, a file in that directory whose filename is the same as the filename of *file2* is used (and vice versa).

DESCRIPTION

There are several options for output format; the default output format contains lines of these forms:

n1 a *n3,n4*
n1,n2 d *n3*
n1,n2 c *n3,n4*

These lines resemble *ed* commands to convert *file1* into *file2*. The numbers after the letters pertain to *file2*. In fact, by exchanging 'a' for 'd' and reading backward one can ascertain equally how to convert *file2* into *file1*. As in *ed*, identical pairs where *n1* = *n2* or *n3* = *n4* are abbreviated as a single number.

Following each of these lines are all of the lines that are affected in the first file flagged by left angle bracket, then all the lines that are affected in the second file flagged by right angle bracket.

DIAGNOSTICS

- Exit status is 0 for no differences, 1 for some, 2 for trouble.

NOTES

- Editing scripts produced under the –e or –f option are naive about creating lines consisting of a single period.
- When comparing directories with the –b, –w or –i options specified, diff first compares the files ala cmp, and then decides to run the diff algorithm if they are not equal. This can cause a small amount of spurious output if the files then turn out to be identical because the only differences are insignificant blank string differences.

FILES

/tmp/d?????	
/usr/lib/diffh	for –h
/bin/diff	for directory diffs
/bin/pr	

SEE ALSO

cmp, cc, comm, ed, diff3

diff3

three-way differential file comparison

The diff3 command compares three versions of a file, and shows disagreeing ranges of text flagged with these codes:

====	all three files differ
====1	*file1* is different
====2	*file2* is different
====3	*file3* is different

---

diff3 [-ex3EX] *file1 file2 file3*

ARGUMENTS

-e creates a script for the editor *ed* that incorporates into *file1* all changes between *file2* and *file3*, *i.e.* the changes that normally would be flagged ==== and ====3.

-x produces a script to incorporate only changes flagged ====.

-3 produces a script to incorporate only changes flagged ====3.

-E similar to -e but overlapping changes (those marked ==== in the normal listing) are treated differently. The overlapping lines from both files will be inserted by the edit script, bracketed with "<<<<<<" and ">>>>>>".

-X similar to -x, but all lines found will be inserted by the edit script bracketed with "<<<<<<" and ">>>>>>".

DESCRIPTION

The type of change suffered in converting a given range of a given file to some other is indicated in one of these ways:

f : *n1* a Text is to be appended after line number *n1* in file *f*, where *f* = 1, 2, or 3.

f : *n1* , *n2* c Text is to be changed in the range line *n1* to line *n2*. If *n1* = *n2*, the range can be abbreviated to *n1*.

The original contents of the range follows immediately after a c indication. When the contents of two files are identical, the contents of the lower-numbered file is suppressed.

The following command applies the resulting script to *file1*.

```
(cat script; echo '1,$p') | ed - file1
```

The -E option is used by RCS merge to ensure that overlapping changes in the merged file are preserved and brought to someone's attention.

NOTES

- Text lines that consist of a single period defeat –e.

FILES

/tmp/d3?????
/usr/lib/diff3

SEE ALSO

diff

dired

directory editor

Dired makes it easy to delete many of the files in a single directory at once. It presents a copy of a listing of the directory, which you can move around in, marking files for deletion. When you are satisfied, you can tell dired to go ahead and delete the marked files. Dired can be invoked either from emacs or directly from the shell.

---

dired [-[sr] [fsrw]] [*directory*]

ARGUMENTS

-sr specifies a sorting sequence: **s** = forward; **r** = reverse.

fsrw specifies the field on which to sort: **f** = filename (default), **s** = size, **r** = read date, **w** = write date.

directory specifies the name of a directory. The default is the current directory.

For example, the following command line reads in your home directory and sorts it in reverse order by size, largest files first.

```
% dired -rs ~
```

DESCRIPTION

Dired can delete files that contain unprintable characters, something you cannot do easily from the shell. However, dired cannot be used to delete files whose names contain embedded spaces; do not try to use it to delete such files.

Basic Dired Commands

To mark a file for deletion, move to the line describing the file and typing D, C-D, K, or C-K. The deletion mark is visible as a D at the beginning of the line. Point is moved to the beginning of the next line, so that several D's delete several files. Alternatively, if you give D an argument, it marks that many consecutive files. Given a negative argument, it marks the preceding file (or several files) and puts point at the first (in the buffer) line marked. Many of the dired commands (D, U, Space, Backspace, Delete) repeat this way with numeric arguments.

To remove a deletion mark, use the U (for Undelete) command, which is invoked like D: it removes the deletion mark from the current line (or next few lines, if given an argument). The Delete command removes the deletion mark from the previous line, moving up to that line. Thus, a Delete after a D precisely cancels the D.

For extra convenience, Space is made a command similar to C-N in emacs. Moving down a line is done so often in dired that it deserves to be easy to type. Delete is often

useful simply for moving up. Other commands for moving down are +, Linefeed, and Return; other commands for moving up are Backspace, –, and ^.

To have the directory sorted in a different way than you see it, type either S for a forward sort or R for a reverse sort, followed by a letter signifying the field you want sorted. The valid letters at this point are F for filename (the default), S for size, R for read date, W for write date, CTRL G to quit out of the command, and ? for a brief explanation. The normal sort order is alphabetical for filenames, increasing number of blocks for size, and most recent first for read and write dates. The reverse sort reverses all of these. When you use one of the sort commands, the directory is read in from scratch, so any D marks you may have made are lost.

If you are not sure whether you want to delete a file, you can examine it by typing E. *Emacs* enters a recursive editing mode on the file, which you can exit with C-M-Z. The file is actually visited at that time, and you can change it if you want. When you exit the recursive editing level, you return to dired. The V command is like E but uses View File to look at the file. If you use the V command on a directory, it displays the directory as in the C-X C-D command of emacs.

If you use the E command on a directory instead of a file, dired proceeds to edit the directory you are pointing to. If you have marked any files for deletion in the current directory, it first confirms that you want to have them deleted, and then go ahead and delete them. If you have given the E command a positive argument, the new directory is sorted using the S command; if you have given it a negative argument, the new directory is sorted using the R command. In both cases, dired prompts you for the type of sort (F, S, R, or W).

If you are in two-window mode when you use the E command on a file, the file is visited in the other window. You can enter two-window mode with the 2 command, and return to one-window mode with the 1 command. See the *emacs* documentation for more information about windows.

It is possible to use dired to change the mode of a file. Position the cursor over the mode bit you want to change, and type A to turn it on (add access) or X to turn it off (deny access). If you type A when you are positioned over an "execute" bit that is already turned on, it is turned into a setuid, setgid, or sticky bit, as appropriate. Similarly, if you are positioned over a setuid, setgid, or sticky bit and type X, it is turned into an "execute" bit. Typing another X while positioned over this bit clears it completely. See chmod for a discussion of the protection bits.

Whenever you set or clear a protection bit, an M appears in the second column of the line if one was not there already. This indicates to *dired* that the mode has been changed. If you change the mode of the file, and then change your mind and decide you want the original mode, you can just type X while positioned over this M. This deletes the M, and dired does not think that the mode has been changed. If you change your mind again, just type A in the second position of the line, and the M is restored. This only works if you have changed the mode of the file at some point. Alternatively, you can just set or clear one or more of the protection bits, and the M reappears.

When you set or clear a protection bit, the cursor automatically moves over to the next column. As in emacs, you can use CTRL F to move forward a column without changing anything and CTRL B to move backward a column. Moving forward is so useful that it is also available as F, in addition to CTRL F. Two other very useful commands in dired are CTRL A, to move to the beginning of the line, and CTRL E, to move to the end of the line.

When you have marked the files you wish to delete or change, you can exit dired with Q or C-M-Z. If any files were marked for deletion or mode change, dired asks you if it should go ahead and make the changes permanent. You can answer Y for yes, N for no, or CTRL G to abort the exiting process. No other inputs are allowed.If you have marked a directory for deletion instead of just a regular file, at this point dired asks you if you really want to delete it. If you answer yes, the directory is deleted, along with all of its files, subdirectories, etc.

Other Dired Commands

B automatically marks all backup files (those ending with "~") with D's.

M moves the file to any other directory or device. You must say where to move it.

H helps you clean up by marking all backup and temporary files with D's. It is just a combination of the B and T commands.

N reads in a new directory. If you give it a positive argument, it also does a sort with the S command; if you give it a negative argument, it does a reverse sort with the R command. In both cases, it prompts you for F, S, R, or W.

P prints the file pointed to on the line printer.

T automatically marks all temporary files (those beginning with "#") with D's.

? displays a list of the dired commands.

Editing the Dired Buffer Yourself

It is unwise to try to edit the text of the directory listing yourself, without using the special dired commands, unless you know what you are doing, since you can confuse dired that way. To make it less likely that you will do so accidentally, the self-inserting characters are all made invalid inside dired. However, deleting whole lines at a time is certainly safe. This does not delete the files described by those lines; instead, it makes dired forget that they are there and thus makes sure they are NOT deleted. Thus, `M-X Keep Lines$FOO<cr>` is useful if you wish to delete only files with a FOO in their names. As another example, if you wanted to get rid of all filenames starting with a period from the display, you could type `M-X Flush Lines$`. See the emacs documentation for more information on `M-X Keep Lines` and `M-X Flush Lines`.

FILES

/usr/lib/emacs/dirhelp

SEE ALSO

emacs, chmod
GNU Emacs Manual

dirs

display directory stack (C shell)

The dirs command displays the directory stack that is manipulated by *pushd* and *popd*.

---

dirs [-l]

ARGUMENTS

-l displays the stack entries in expanded format.

DESCRIPTION

The top of the stack (the current working directory) is to the left.

NOTES

- Dirs does not work properly once you have crossed through a symbolic link.

SEE ALSO

pushd, popd

domainname

set or display name of current domain system

The domainname command sets or displays the name of the current domain. The yellow pages use the domain to refer to a group of hosts.

---

domainname [*name*]

ARGUMENTS

name specifies a new domainname. Only the superuser can set the domainname; this is usually done in the startup script /etc/rc.local.

DESCRIPTION

With no arguments, domainname displays the name of the current domain.

SEE ALSO

Concentrix-2800 System Reference Manual: System Calls and C Library Subroutines: getdomainname, setdomainname
Concentrix-2800 System Administration: ypinit

du

summarize disk usage

The du command displays the number of kilobytes contained in all files and, recursively, directories within each specified directory or file. By default, du generates an entry for each directory only.

---

du [-s] [-a] [*name...*]

ARGUMENTS

-s causes only the grand total to be given.

-a causes an entry to be generated for each file.

name specifies the name of a file or directory. If omitted, the current working directory is used.

DESCRIPTION

On a 4.3BSD file system, du reports the number of kiloybytes used in multiples of 1KB. On a tfs file system, it reports the number of kilobytes used in multiples of 4KB.

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

NOTES

- A file may actually reserve or use more filesystem space than the number of bytes that it contains.
- Non-directories given as arguments (not under -a option) are not listed.
- If there are too many distinct linked files, du counts the excess files multiply.
- Du ignores symbolic links.

SEE ALSO

df

echo

echo arguments

The echo command displays its arguments on the standard output, separated by blanks and terminated by a newline.

---

echo [-n] [*arg*...]

ARGUMENTS

-n suppresses the newline.

arg can be a quoted string, an ambiguous file reference, a shell variable, and so forth.

DESCRIPTION

Echo is useful for producing diagnostics in shell programs and for writing constant data on pipes.

In the Bourne shell, you can send messages to the standard error file by writing:

echo *message* **1>&2**

ed

text editor

The ed command invokes the text editor ed.

---

ed [–] [–x] [*name*]

ARGUMENTS

– suppresses explanatory output and should be used when the standard input is an editor script.

–x simulates an *x* command first to handle an encrypted file.

name simulates an *e* command on the named file; that is, the file is read into ed's buffer so that it can be edited.

DESCRIPTION

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

NOTES

- The l command mishandles DEL.
- The undo command causes marks to be lost on affected lines.

DIAGNOSTICS

- ?name for inaccessible file
- ?self-explanatory message for other errors.
- To protect against throwing away valuable work, a *q* or *e* command is considered to be in error, unless a *w* has occurred since the last buffer change. A second *q* or *e* are obeyed regardless.

FILES

/tmp/e*
edhup: work is saved here if terminal hangs up

SEE ALSO

edit, ex, sed, crypt
Concentrix-2800 System User's Guide: Line Editing (ed) Tutorial
Concentrix-2800 System User's Guide: Line Editing (ed) Advanced

edit

simple text editor

The edit command invokes edit, a text editor based on ex. Edit avoids some of the complex features used mostly by systems programmers and persons very familiar with ed. If you have not used ed, or are a casual user, you may find that edit is useful for you.

---

edit [*option...*] *name...*

ARGUMENTS

-	suppresses all interactive user feedback and is useful in processing editor scripts in command files.
-v	is equivalent to using *vi* rather than *ex*.
-t *tag*	is equivalent to an initial tag command, editing the file containing the *tag* and positioning the editor at its definition.
-r	recovers after an editor or system crash, retrieving the last saved version of the named file or, if no file is specified, typing a list of saved files.
-l	sets up for editing Lisp programs, setting the *showmatch* and *lisp* options.
-w *n*	set the default window size to *n* and is useful on dialups to start in small windows.
-x	causes ex to prompt for a *key*, which is used to encrypt and decrypt the contents of the file, which should already be encrypted with the same key.
-R	sets the *readonly* option at the start.
+*command*	indicates that the editor should begin by executing the specified *command*. If *command* is omitted, it defaults to dollar sign, positioning the editor at the last line of the first file.
name	is the name of a file to be edited.

NOTES

- The Concentrix-2800 command e is a hard link to edit, and can be used in its place.
- The undo command causes all marks to be lost on lines changed and then restored if the marked lines were changed.
- Undo never clears the buffer modified condition.

- The z command displays a number of logical rather than physical lines. More than a screen full of output can result if long lines are present.
- File input/output errors do not display a name if the command line hyphen option is used.
- There is no easy way to do a single scan ignoring case.
- The editor does not warn if text is placed in named buffers and not used before exiting the editor.
- Null characters are discarded in input files, and cannot appear in resultant files.

FILES

/usr/lib/ex?.?strings	error messages
/usr/lib/ex?.?recover	recover command
/usr/lib/ex?.?preserve	preserve command
/etc/termcap	describes capabilities of terminals
~/.exrc	editor startup file
/tmp/Ex*nnnnn*	editor temporary
/tmp/Rx*nnnnn*	named buffer temporary
/usr/preserve	preservation directory

SEE ALSO

awk, ed, ex, grep, sed, grep, vi
Concentrix-2800 System User's Guide: Line Editing (edit) Tutorial
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: termcap

egrep

search file for regular expression

The egrep command searches files for lines matching a pattern. Normally, each line found is copied to the standard output. egrep patterns are full regular expressions; it uses a fast deterministic algorithm that sometimes needs exponential space.

---

egrep [*option...*] *expression* [*file...*]

ARGUMENTS

-v displays all lines but those matching.

-c displays only a count of matching lines.

-l displays the names of files with matching lines (once) separated by newlines.

-n precedes each line by its relative line number in the file.

-b precedes each line by the block number on which it was found (sometimes useful in locating disk block numbers by context).

-s displays nothing except error messages (useful for checking the error status).

-e *expression*
is the same as an *expression* argument, but useful when the *expression* begins with a hyphen.

-f *file* the regular expression is taken from the specified *file*.

expression
is an extended regular expression as described below.

file specifies a file to search. The default is the standard input.

In all cases the file name is shown if there is more than one input file. Care should be taken when using the characters dollar sign, asterisk, left bracket, circumflex, vertical bar, parenthesis, and backslash in the *expression* as they are also meaningful to the Shell. It is safest to enclose the entire *expression* argument in single quotes.

DESCRIPTION

egrep accepts extended regular expressions. In this section, 'character' excludes newline:

- A backslash (\) followed by a single character other than newline matches that character.

- The circumflex character (^) matches the beginning of a line.
- The dollar sign character ($) matches the end of a line.
- A period (.) matches any character.
- A single character not otherwise endowed with special meaning matches that character.
- A string enclosed in brackets ([]) matches any single character from the string. Ranges of ASCII character codes can be abbreviated as in 'a–z0–9'. A bracket can occur only as the first character of the string. A literal hyphen (–) must be placed where it can't be mistaken as a range indicator.
- A regular expression followed by an asterisk (*) matches a sequence of zero or more matches of the regular expression. A regular expression followed by a plus sign (+) matches a sequence of one or more matches of the regular expression. A regular expression followed by a question mark (?) matches a sequence of zero or one matches of the regular expression.
- Two regular expressions concatenated match a match of the first followed by a match of the second.
- Two regular expressions separated by a vertical bar (|) or newline match either a match for the first or a match for the second.
- A regular expression enclosed in parentheses matches a match for the regular expression.

The order of precedence of operators at the same parenthesis level is brackets, then asterisk, plus sign, and question mark, then concatenation, then vertical bar and newline.

DIAGNOSTICS

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

NOTES

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

SEE ALSO

grep, fgrep, ex, sed, sh

else

alternate branch in if-then-else (C shell)

See the description of the **if-then-else** command.

emacs

screen oriented display editor

GNU Emacs is a display-oriented real-time text editor.

---

emacs [*option*] [[+*linemum*] *file*]... [**-l** *file*][**-f** *function*][**-i** *file*][**-kill**]

ARGUMENTS

file visits the specified files in successive buffers (find file function). You are left in the last buffer visited.

+*linenum*
starts at line number *linenum* in *file*.

-i inserts the contents of *file* into the current buffer. (insert-file function)

-l loads a file *file* of Lisp code with the function load. (load-file function.)

-load same as -l.

-f calls Lisp function *function* with no arguments.

-funcall
calls Lisp function *function* with no arguments.

-kill exits from GNU Emacs without confirmation.

The following options are only recognized at the beginning of the command line. If more than one appears, they must appear in the order listed.

-t *device*
Use *device* as the device for terminal input and output.

-d *display*
When running with the X window system, use the display named *display* to make the window that serves as the GNU Emacs terminal.

-batch Run GNU Emacs in *batch mode*, which means that text being edited is not displayed, and standard Unix interrupt characters such as C-z and C-c continue to have their normal effect. Outputs to stdout only what would normally be printed in the echo area under program control.

Batch mode is used for running programs written in GNU Emacs Lisp from shell scripts, makefiles, and so on. The -batch function implies -q. It also causes GNU Emacs to kill itself after all options have been processed. In addition, auto-saving is not done except in buffers for which it is explicitly requested.

-q Do not load your GNU Emacs init file ~/.emacs

-no-init-file
Same as -q.

-u *user* Load user's Emacs init file ~/.emacs instead of your own.

-user *user*
same as -u.

DESCRIPTION

The *GNU Emacs Manual* provides full details on using GNU Emacs. Within GNU Emacs, typing CTRL h gives access to the comprehensive GNU Emacs on-line help facility. Help Tutorial (CTRL h t) requests an interactive tutorial which can teach beginners the fundamentals of GNU Emacs in a few minutes.

In GNU Emacs, as soon as you type any character, some action is performed and you see the resulting buffer. Generally, graphic, printing characters insert themselves, while control characters do the editing. So, to insert some text, just type it. To make corrections, use the control characters. Here is a simple description of what various control characters do:

C-f	Move forward one character.
C-b	Move backward one character.
Backspace	Move backward one character (same as C-B).
C-d	Delete next character.
Rubout	Delete last character. (Rubout isn't a control character, but its function is like those of control characters).
C-a	Move to beginning of this line.
C-e	Move to end of this line.
C-n	Move to next line.
C-p	Move to previous line.
C-k	Kill rest of line.
C-y	Un-kill (yank) what was just killed, inserting it into the buffer at the current position.
C-@	Mark this place in the buffer.
C-w	Kill (wipe out) from here to marked place. (C-y unkills C-w-killed text too.)
C-l	Clear screen, redisplay buffer.
C-g	(Quit) Stops whatever EMACS is doing.
C-x	Prefix character: follow with another character to make a two-character C-x command.
C-x C-z	Exit from GNU EMACS.
C-x C-v	Visit a file. See below.
C-x C-s	Save the file you are visiting.

An easy way to move text around is to kill it, move it, then un-kill it. You can un-kill text any number of times to create copies in different places.

To edit an already existing file, type the C-x C-v command, followed by the name of the file you would like to edit. When you have finished typing the filename, end it with a RETURN. The file's name appears in the line that starts with "Emacs " (near the bottom). Now you are ready to edit the text of the file.

After you have typed for a while, type C-x C-s to save what you have added to the file so far. The text does not really go into the file until you save it. After saving, you can edit some more and save again. If the system crashes, you do not lose anything that you have saved.

To switch to writing a different file, just use the C-x C-v command again. The same command allows you to make changes in an old file. If you do C-x C-v when you have made changes to a file but not saved them, GNU Emacs asks you whether to save them. Type either y or n as the answer, or type C-g to quit, look around to decide what to do, save or not, and repeat the C-x C-v. Don't ever be afraid to quit.

USING GNU EMACS WITH THE X WINDOW SYSTEM

GNU Emacs has been tailored to work well with the X window system. To enable this feature, you must define the macro HAVE_X_WINDOWS in the file src/config.h before compiling GNU Emacs. If you run GNU Emacs from under X windows, it will create its own X window to display in. You will probably want to start the editor as a background process so that you can continue using your original window. To use the optional X Menu features, define the macro HAVE_X_MENU. This macro is separate from HAVE_X_WINDOWS because the Menu facility of X does not work on all the systems that support X. GNU Emacs can be started with the following X options:

-r Display the Emacs window in inverse video.

-i Use the "kitchen sink" bitmap icon when iconifying the GNU Emacs window.

-font *font*
Set the GNU Emacs window's font to that specified by *font*. You will find the various X fonts in the /usr/new/lib/X/font directory. Note that Emacs will only accept fixed-width fonts. These include the 6x10.onx, 6x13.onx, 6x13p.onx, 8x13.onx, and 9x15.onx fonts. The other fixed-width fonts are specified by the fx character sequence that comes before the .onx extension. When you specify a font, do not include the .onx extension. Be sure to put a space between the -font option and the *font* specification argument.

-b *pixels*
Set the GNU Emacs window's border width to the number of pixels specified by *pixels*.

-ib *pixels*
Set the window's internal border width to the number of pixels specified by *pixels*. Defaults to one pixel of padding on each side of the window.

-w =[WIDTH][xHEIGHT][{+-}XOFF[{+-}YOFF]]
Set the GNU Emacs window's width, height, and position on the screen. [ ] denotes optional arguments, { } surrounds alternatives. WIDTH and HEIGHT are in number of characters; XOFF and YOFF are in pixels. WIDTH defaults to 80, HEIGHT to 24, XOFF and YOFF to 1. If you don't give XOFF and/or YOFF, then you must use the mouse to create the window. If you give XOFF and/or YOFF, then a WIDTHxHEIGHT window will automatically be created. XOFF and YOFF specify deltas from a corner of the screen to the corresponding corner of the window, as follows:

+XOFF+YOFF	upper left to upper left
-XOFF+YOFF	upper right to upper right
+XOFF-YOFF	lower left to lower left
-XOFF-YOFF	lower right to lower right

-fg *color*
On color displays, sets the color of the text.

-bg *color*
On color displays, sets the color of the window's background. See the file /usr/lib/rgb.txt for a list of valid color names.

-bd *color*
On color displays, sets the color of the window's border. See the file /usr/lib/rgb.txt for a list of valid color names.

-cr *color*
On color displays, sets the color of the window's text cursor. See the file /usr/lib/rgb.txt for a list of valid color names.

-ms *color*
On color displays, sets the color of the window's mouse cursor. See the file /usr/lib/rgb.txt for a list of valid color names.

-d *displayname*
Create the Emacs window on the display specified by *displayname*. Must be the first option specified in the command line.

-nw tells GNU Emacs not to use its special interface to X. If you use this option when invoking GNU Emacs from an xterm window, display is done in the xterm window. This must be the first option specified in the command line.

Setting X Default Values for GNU Emacs Windows

You can set X default values for your GNU Emacs windows in your .Xdefaults file. Use the following format:

emacs.keyword:value

where value specifies the default value of keyword. GNU Emacs lets you set default values for the following keywords:

BodyFont
Sets the window's text font.

ReverseVideo
If ReverseVideo's value is set to on, the window will be displayed in inverse video.

BitMapIcon
If BitMapIcon's value is set to on, the window will iconify into the "kitchen sink."

BorderWidth
Sets the window's border width in pixels.

Foreground
For color displays, sets the window's text color. See the file /usr/lib/rgb.txt for a list of valid color names.

Background
: For color displays, sets the window's background color. See the file /usr/lib/rgb.txt for a list of valid color names.

Border
: For color displays, sets the color of the window's border. See the file /usr/lib/rgb.txt for a list of valid color names.

Cursor
: For color displays, sets the color of the window's text cursor. See the file /usr/lib/rgb.txt for a list of valid color names.

Mouse
: For color displays, sets the color of the window's mouse cursor. See the file /usr/lib/rgb.txt for a list of valid color names.

If you try to set color values while using a black and white display, the window's characteristics will default as follows: the foreground color will be set to black, the background color will be set to white, the border color will be set to grey, and the text and mouse cursors will be set to black.

Using the Mouse

The following table lists the key bindings for the mouse cursor when used in a GNU Emacs window.

MOUSE BUTTON	FUNCTION
left	set mark
middle	set cursor
right	select (GNU Emacs) window
SHIFT-middle	put text into X cut buffer (cut text)
SHIFT-right	paste text
CTRL-middle	cut text and kill it
CTRL-right	select this window, then split it into two windows
CTRL-SHIFT-left	X buffer menu--hold the buttons and keys down, wait for menu to appear, select buffer, and release. Move mouse out of menu and release to cancel.
CTRL-SHIFT-middle	X help menu--pop up menu for Emacs help.
CTRL-SHIFT-right	Select window with mouse, and delete all other windows. Same as typing CTRL x 1.

BUGS

Mailing list for reporting bugs and fixes: bug-gnu-emacs@prep.ai.mit.edu, on the internet (ucbvax!prep.ai.mit.edu!bug-gnu-emacs on UUCPnet). But before reporting something as a bug, please check that it really is a bug, not a misunderstanding nor a deliberate feature. Read the section "Reporting Emacs Bugs" near the end of the reference manual (or Info system) for hints on how and when to report bugs. Also, include the version number of the GNU Emacs you are running in every bug report.

Do not expect a personal answer to a bug report. The purpose of reporting bugs is to get them fixed for everyone in the next release, if possible. For personal assistance, look in the SERVICE file (see above) for a list of people who offer it.

Please do not send anything but bug reports to this mailing list. Send requests to be added to mailing lists to the special list info-gnu-emacs-request@prep.ai.mit.edu (or the corresponding UUCP address). For more information about GNU Emacs mailing lists, see the file /usr/local/emacs/etc/MAILINGLISTS. If bugs can be isolated, they get fixed, so it is in your interest to report them in such a way that they can be easily reproduced.

Known bugs: shell does not work with programs running in raw mode on some Unix versions.

FILES

/tmp/????????????
/usr/lib/emacs/*
/usr/preserve/emdir*
/usr/preserve/emkeys*
/usr/temp/emsav*
~/.emacs

/usr/local/emacs/src
/usr/local/emacs/man
/usr/local/emacs/etc

/usr/local/emacs/lisp – Lisp source files and compiled files that define most editing commands. Some are preloaded; others are autoloaded from this directory when used.

/usr/local/emacs/etc/DOC.*
contains the documentation strings for the Lisp primitives and preloaded Lisp functions of GNU Emacs. They are stored here to reduce the size of Emacs proper.

/usr/local/emacs/etc/DIFF discusses GNU Emacs vs.Twenex Emacs;
/usr/local/emacs/etc/CCADIFF discusses GNU Emacs vs.CCA Emacs;
/usr/local/emacs/etc/GOSDIFF discusses GNU Emacs vs.Gosling Emacs.

/usr/local/emacs/etc/SERVICE
lists people offering various services to assist users of GNU Emacs, including education, troubleshooting, porting and customization. These files also have information useful to anyone wishing to write programs in the GNU Emacs Lisp extension language, which has not yet been fully documented.

/usr/local/emacs/info
: files for the Info documentation browser (a subsystem of Emacs).The complete text of the Emacs reference manual is included in a convenient tree structured form.

/usr/local/emacs/lock
: holds lock files made for all files being modified in GNU Emacs, to prevent simultaneous modification of one file by two users.

/usr/local/emacs/cpp
: the GNU cpp, needed for building Emacs on certain versions of Unix where the standard cpp cannot handle long names for macros.

UNRESTRICTIONS

GNU Emacs is free; anyone may redistribute copies of GNU Emacs to anyone under the terms stated in the GNU Emacs General Public License, a copy of which accompanies each copy of GNU Emacs and which also appears in the reference manual.

Copies of GNU Emacs may sometimes be received packaged with distributions of Unix systems, but it is never included in the scope of any license covering those systems. Such inclusion violates the terms on which distribution is permitted. In fact, the primary purpose of the General Public License is to prohibit anyone from attaching any other restrictions to redistribution of GNU Emacs.

Richard Stallman encourages you to improve and extend GNU Emacs, and urges that you contribute your extensions to the GNU library. Eventually GNU (Gnu's Not Unix) will be a complete replacement for Berkeley Unix. Everyone will be able to use the GNU system for free.

GNU Emacs was written by Richard Stallman and the Free Software Foundation. Joachim Martillo and Robert Krawitz added the X features.

SEE ALSO

GNU Emacs Manual

end

terminate control structure (C shell)

See the descriptions of the foreach, if, switch, and while commands.

endif

terminate if-then-else (C shell)

See the description of the if-then-else command.

endsw

terminate switch (C shell)

See the description of the **switch** command.

enroll

enter password for encrypted mail

The enroll command asks for and stores a password that you must subsequently use to receive encrypted (secret) mail.

---

enroll

DESCRIPTION

Enroll, along with xget and xsend, implement a secure communication channel similar to mail except that no one can read the messages except the intended recipient. The method embodies a public-key cryptosystem using knapsacks.

NOTES

- Encrypted mail should be integrated with ordinary mail.
- The announcement of encrypted mail makes traffic analysis possible.

FILES

/usr/spool/secretmail/*.key	keys
/usr/spool/secretmail/*.[0-9]	messages

SEE ALSO

xget, xsend, mail

eqn

mathematics preprocessor for troff (unsupported)

The eqn command invokes a troff preprocessor for typesetting mathematics on a Graphic Systems phototypesetter.

---

eqn [-d*xy*] [-p*n*] [-s*n*] [-f*n*] [*file...*]

ARGUMENTS

-d*xy* specifies that the characters *x* and *y* are equation delimiters. The left and right delimiters can be identical.

-p*n* controls the point size of subscripts and superscripts (reduced by three point sizes from the previous size by default).

-s*n* controls type size.

-f*n* controls the font.

file if no files are specified, the program reads from the standard input.

NOTES

- To embolden digits, parens, etc., enclose them in double quotes, for example: `bold "12.3"`.

SEE ALSO

neqn, checkeq, troff, tbl
Document Preparation Guide: EQN – Mathematics Typesetting
Document Preparation Guide: Using NROFF/TROFF

error

analyze/disperse compiler error messages

The error command analyzes and optionally disperses the diagnostic error messages produced by a number of compilers and language processors to the source file and line where the errors occurred. It can replace the traditional method of writing abbreviations of errors on paper, and permits error messages and source code to be viewed simultaneously without machinations of multiple windows in a screen editor.

NOTE: Error does not work with FX/Fortran-2800.

---

error [*option...*] [*name*]

ARGUMENTS

-n suppresses touching of files; sends all error messages to the standard output.

-s displays statistics regarding the error categorization.

-q asks for permission to touch a file. A `y` or `n` to the question is necessary to continue. Absence of the -q option implies that all referenced files (except those referring to discarded error messages) are to be touched.

-v after all files have been touched, overlays the editor vi set up to edit all files touched, and positioned in the first touched file at the first error. If vi can't be found, error tries ex or ed from standard places.

-t*list* takes the argument *list* as a suffix list. Files whose suffixes do not appear in the suffix list are not touched. The suffix list is dot separated, and asterisk wildcards work. Thus the suffix list:

```
".c.y.foo*.h"
```

allows error to touch files ending with `.c`, `.y`, `.foo*` and `.y`.

-I*name* takes the names of functions to ignore either from the file named `.errorrc` in your home directory, or from the file named by the -I option.

name specifies a file containing error messages. Error is intended to be run with its standard input connected via a pipe to the error message source. Some language processors put error messages on their standard error file; others put their messages on the standard output. Thus, both error sources should be piped together into *error*. For example, when using the *csh* syntax,

```
make -s lint |& error -q -v
```

analyzes all the error messages produced by whatever programs make runs when making lint.

DESCRIPTION

Error looks at the error messages, either from the specified file *name* or from the standard input, and attempts to determine which language processor produced each error message, determines the source file and line number to which the error message refers, determines if the error message is to be ignored or not, and inserts the (possibly slightly touched) error message into the source file as a comment on the line preceding to which the line the error message refers. Error messages that can't be categorized by language processor or content are not inserted into any file, but are sent to the standard output. *Error* touches source files only after all input has been read.

By specifying the -q query option, you are asked to confirm any potentially dangerous (such as touching a file) or verbose action. If the -t option and associated suffix list is given, error restricts itself to touch only those files with suffices in the suffix list. Error also can be asked (by specifying -v) to invoke vi on the files in which error messages were inserted; this obviates the need to remember the names of the files with errors.

Error knows about the error messages produced by: make, cc, cpp, ccom, as, ld, lint, and pc. Error knows a standard format for error messages produced by the language processors, so is sensitive to changes in these formats. For all languages except Pascal, error messages are restricted to be on one line. Some error messages refer to more than one line in more than one file; error duplicates the error message and inserts it at all of the places referenced.

Error does one of six things with error messages.

synchronize Some language processors produce short errors describing which file it is processing. Error uses these to determine the file name for languages that don't include the file name in each error message. These synchronization messages are consumed entirely by error.

discard Error messages from lint that refer to one of the two lint libraries, /usr/lib/llib-lc and /usr/lib/llib-port are discarded, to prevent accidently touching these libraries. Again, these error messages are consumed entirely by error.

nullify Error messages from lint can be nullified if they refer to a specific function, that is known to generate diagnostics that are not interesting. Nullified error messages are not inserted into the source file, but are written to the standard output. The names of functions to ignore are taken from either the file named .errorrc in the users's home directory, or from the file named by the -l option. If the file does not exist, no error messages are nullified. If the file does exist, there must be one function name per line.

not file specific Error messages that can't be intuited are grouped together, and written to the standard output before any files are touched. They are not inserted into any source file.

file specific Error message that refer to a specific file, but to no specific line, are written to the standard output when that file is touched.

true errors Error messages that can be intuited are candidates for insertion into the file to which they refer.

Only true error messages are candidates for inserting into the file they refer to. Other error messages are consumed entirely by error or are written to the standard output. Error inserts the error messages into the source file on the line preceding the line the language processor found in error. Each error message is turned into a one line comment for the language, and is internally flagged with the string *###* at the beginning of the error, and *%%%* at the end of the error. This makes pattern searching for errors easier with an editor, and allows the messages to be easily removed.

In addition, each error message contains the source line number for the line the message refers to. A reasonably formatted source program can be recompiled with the error messages still in it, without having the error messages themselves cause future errors. For poorly formatted source programs in free format languages, such as C or Pascal, it is possible to insert a comment into another comment, which can wreak havoc with a future compilation. To avoid this, programs with comments and source on the same line should be formatted so that language statements appear before comments.

Error catches interrupt and terminate signals, and if in the insertion phase, orderly terminates what it is doing.

FILES

~/.errorrc	function names to ignore for *lint* error messages
/dev/tty	user's teletype

NOTES

- Error opens the teletype directly to do user querying.
- Source files with links make a new copy of the file with only one link to it.
- Changing a language processor's format of error messages can cause error to not understand the error message.
- Error, since it is purely mechanical, does not filter out subsequent errors caused by "floodgating" initiated by one syntactically trivial error. Humans are still much better at discarding these related errors.
- Pascal error messages belong after the lines affected (error puts them before). The alignment of the vertical bar marking the point of error is also disturbed by error.
- Error was designed for work on video terminals at reasonably high speed. It is less pleasant on slow speed terminals, and has never been used on hardcopy terminals.

eval

evaluate arguments and execute resulting comand

The eval command reads its arguments as input to the shell and executes the resulting command(s) in the context of the current shell. It is usually used in shell scripts to execute commands generated as the result of command or variable substitution, since parsing occurs before the substitutions.

---

eval [*arg...*]

ARGUMENTS

arg is an expression that forms one or more valid commands.

NOTES

- The eval command is built into both the C shell and Bourne shell. The two versions are functionally equivalent.

ex

text editor

The ex command invokes ex, the root of a family of text editors: edit, ex and vi.

---

ex [*option...*] *name...*

ARGUMENTS

- suppresses all interactive user feedback and is useful in processing editor scripts in command files.

-v is equivalent to using *vi* rather than *ex*.

-t *tag* is equivalent to an initial *tag* command, editing the file containing the *tag* and positioning the editor at its definition.

-r recovers after an editor or system crash, retrieving the last saved version of the named file or, if no file is specified, typing a list of saved files.

-l sets up for editing Lisp programs, setting the *showmatch* and *lisp* options.

-w *n* set the default window size to *n* and is useful on dialups to start in small windows.

-x causes *ex* to prompt for a *key*, which is used to encrypt and decrypt the contents of the file, which should already be encrypted with the same key.

-R sets the *readonly* option at the start.

+*cmd* indicates that the editor should begin by executing the specified command. If *cmd* is omitted, it defaults to dollar sign, positioning the editor at the last line of the first file.

name is the name of a file to be edited.

FILES

/usr/lib/ex?.?strings	error messages
/usr/lib/ex?.?recover	recover command
/usr/lib/ex?.?preserve	preserve command
/etc/termcap	describes capabilities of terminals
~/.exrc	editor startup file
/tmp/Ex*nnnnn*	editor temporary
/tmp/Rx*nnnnn*	named buffer temporary
/usr/preserve	preservation directory

SEE ALSO

Concentrix-2800 System User's Guide: Line Editing (ex) Reference
Concentrix-2800 System User's Guide: Display Editing (vi)
Concentrix-2800 System User's Guide: Line Editing (ed) Tutorial
Concentrix-2800 System User's Guide: Line Editing (ed) Advanced
(The material in the ed documents also works with ex.)

exec

execute command in place of current shell (C shell)

The exec command executes the specified command in place of the current shell without creating a new process. It is intended for use in shell scripts.

---

exec *arg...*

ARGUMENTS

arg is a command and optional input/output arguments.

NOTES

- The exec command is built into both the C shell and Bourne shell. In the C shell version, you must supply a command.

SEE ALSO

execute
Concentrix-2800 System Reference: System Calls and C Library Subroutines: execl, execve, execven

exec

execute command in place of current shell (Bourne shell)

The exec command executes the specified command in place of the current shell without creating a new process. It is intended for use in shell scripts.

---

exec *arg...*

ARGUMENTS

arg is a command and optional input/output arguments.

NOTES

- The exec command is built into both the C shell and Bourne shell. In the Bourne shell version, you can omit the command and supply only input/output arguments, causing the shell's input/output to be modified.

SEE ALSO

execute
Concentrix-2800 System Reference: System Calls and C Library Subroutines: execl, execve, execven

execute

run program on specific processor type

The execute command allows you to specify the type of computing resource on which a program is to run.

---

execute [*option ...*] *command ...*

ARGUMENTS

-ac forces the use of the activated coherency cache (ACC) on a Model 200 processor module.This option can be used in conjunction with the -w option to add the ELF note forceACcache to an executable or object file. The -ac and -nac options are mutually exclusive - they should not be specified together, and adding one as an ELF note removes the ELF note for the other.

-ce forces *command* to run as a detached process. This option can be used in conjunction with the -w option to add the ELF note nocluster to an executable or object file.

Note: This does not disallow the process from running on an IOM or a cluster, but it will run as a detached process. Use a special scheduling class to force a process onto a particular resource or resource type.

-c[*n*] forces *command* to run on the computational complex. The *n* argument is an integer that specifies the number (in the range 1 to 14) of CEs to apply to the *command* and is normally used for benchmarking concurrent programs. Any unused CEs in the complex remain idle while *command* is running. If you omit *n*, *command* runs on the complex with the maximum possible number of CEs. This option can be used in conjunction with the -w option to add the ELF note clustermax=*n* to an executable or object file.

The user job is scheduled on a cluster with at least *n* processors, if possible. But if not, the job is still scheduled and the following message is written to the user's terminal:

```
Reducing from n processors to m
```

-cl[*n*] forces *command* to run in scheduling class *n* (where *n* is an integer in the range 1 to 32).

-nac disables the use of the activated coherency cache (ACC) on a Model 200 processor module.This option can be used in conjunction with the -w option to add the ELF note disableACcache to an executable or object file. The -ac and -nac options are mutually exclusive - they should not be specified together, and adding one as an ELF note removes the ELF note for the other.

-ndc forces data not to be cached even when a job is run on a detached processor or on a complex of one processor. Since caching improves performance, this option is not normally used. It might be used when doing benchmarking or tuning that attempts to quantify the effect of the number of processors when the test includes the one processor case.

-tmessage_on_*exception=n*
causes a message to be written to the user terminal on every *n*th occurrence of the indicated exception (positive threshold value). See the list of exceptions below.

-tquiet_*exception*
causes no action to be taken when the indicated exception is detected (threshold value of 0). See the list of exceptions below.

-tsignal_on_*exception=n*
causes a signal to be sent to the program on every *n*th occurrence of the indicated exception (negative threshold value). See the list of exceptions below.

-w modifies the header to an object module or executable program with the other specified options on the command line. As a result, subsequent runs of the program will always use the specified options. Running execute with the -w switch marks but does not actually execute a program.

-w can be used with -ac, -nac, -ce, and -c*n*. -w cannot be used with -cl[*n*], and -ndc.

command is an executable file, with arguments, or (if -w is used) an object file.

LIST OF EXCEPTIONS

instruction_access_trap
trap_instruction
missing_unlock
integer_overflow
inexact_result
divide_by_zero
underflow
overflow
operand_error
unordered_compare
denormal
any_floating_point_trap
data_access_trap
unalign_data_access
interrupt

DIAGNOSTICS

- `Not owner` – By default, you must be the superuser to use the -cl option. The system administration command setclassdef all can be specified by the superuser to permit all users to use the -cl option.

EXAMPLES

Force the program to run as a detached process:

```
% execute -ce a.out
```

Use the -w switch so that subsequent invocations of a program will run as a detached process:

```
% execute -w -ce a.out
```

Use the -w switch so that subsequent invocations of a program will disable the ACC on a Model 200 processor module:

```
% execute -w -nac a.out
```

Execute the program on the complex with 4 CEs applied:

```
% execute -w -c4 a.out
```

Execute the program so that a signal is received if the indicated exception occurs:

```
% execute -tsignal_on_divide_by_zero=1 a.out
```

NOTES

- The -ce option overrides any -c# option.
- To time a command issued via execute, use the following syntax:

```
% time execute [option...] command
```

or

```
% /bin/time execute [option...] command
```

SEE ALSO

exec, time, execve, flushicache
Concentrix-2800 System Administration: System Tuning
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File formats: a.out, elf

exit

exit from shell (C shell)

The exit command exits from the current shell. It is intended for use in shell scripts.

---

exit

exit (*expr*)

ARGUMENTS

With no arguments, the shell exits with the value of the `status` variable (the status of the last command executed).

expr is an numeric, filename, or command expression as described in the *Concentrix-2800 System User's Guide, C Shell Reference* whose value specifies the exit status.

SEE ALSO

onintr

exit

exit from shell (Bourne shell)

The exit command exits from a non-interactive shell. It is intended for use in shell scripts.

---

exit (*n*)

ARGUMENTS

n (optional) numeric argument specifying exit status.

NOTES

With no arguments, the shell exits with the value of the `status` variable (the status of the last command executed).

SEE ALSO

onintr

expand

expand tabs to spaces

The expand command processes the named files changing tabs into spaces. Backspace characters are preserved into the output and decrement the column count for tab calculations. Expand is useful for preprocessing character files (before sorting, looking at specific columns, etc.) that contain tabs.

---

expand [*-tab*] [*-tab1,tab2,...*] [*file...*]

ARGUMENTS

-tab if a single *tab* argument is given, tabs are set the specified number of spaces apart instead of the default eight.

-tab1,tab2,...
if multiple tabstops are given, the tabs are set at those specific columns.

file specifies an input file.

SEE ALSO

unexpand

explain

interactive thesaurus for diction

The explain command invokes an interactive thesaurus for the phrases found by diction to be "obscure, awkward, or strange." It attempts to explain why diction found the phrase and suggests alternatives.

---

explain

SEE ALSO

diction
Concentrix Document Preparation Guide: The Style and Diction Programs

expr

evaluate an expression

The expr command evaluates its arguments as an expression and writes its result on the standard output.

---

expr *arg...*

ARGUMENTS

arg each argument is a separate token of the expression.

DESCRIPTION

The operators and keywords are listed below. The list is in order of increasing precedence, with equal precedence operators grouped.

expr | *expr* yields the first *expr* if it is neither null nor 0, otherwise yields the second *expr*.

expr & *expr* yields the first *expr* if neither *expr* is null or 0, otherwise yields '0'.

expr relop expr where *relop is one of* < <= = != >= >, yields 1 if the indicated comparison is true, 0 if false. The comparison is numeric if both *expr* are integers, otherwise lexicographic.

expr + *expr*
expr − *expr* performs addition or subtraction of the arguments.

expr * *expr*
expr / *expr*
expr % *expr* performs multiplication, division, or remainder of the arguments.

expr : *expr* compares the string first argument with the regular expression second argument; regular expression syntax is the same as that of *ed*. The \(...\) pattern symbols can be used to select a portion of the first argument. Otherwise, the matching operator yields the number of characters matched (0 on failure).

(*expr*) parentheses delimit a subexpression.

DIAGNOSTICS

Expr returns the following exit codes:

0 if the expression is neither null nor 0
1 if the expression is null or 0
2 for invalid expressions.

EXAMPLE

The following command adds 1 to the shell variable *a*:

```
a=`expr $a + 1`
```

The following command finds the filename part (least significant part) of the pathname stored in variable *a*, which may or may not contain slash. Note the quoted shell metacharacters.

```
expr $a : '.*/\(.*\)' '|' $a
```

SEE ALSO

sh, test

export

mark names for export (Bourne shell)

The export command marks given names for automatic export to the *environment* of subsequently-executed commands. If no names are given, a list of exportable names is printed.

---

export [*name*]...

ARGUMENTS

name each name listed is marked for export. If no names are given, a list of exportable names is printed.

DESCRIPTION

The environment is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. (See the system call execve.)

On invocation, the shell scans the environment and creates a *parameter* for each name found, giving it the corresponding value. Executed commands inherit the same environment. If the user modifies the values of these *parameters* or creates new ones, none of these affects the environment unless the export command is used to bind the shell's *parameters* to the environment.

EXAMPLE

The following example shows two ways to change the environment of a *simple-command*. First, prefixing assignments to parameters; second, using export:

```
TERM=450 cmd args

(export TERM; TERM=450; cmd args)
```

SEE ALSO

exec, execven, sh
Concentrix-2800 System User's Guide: Using the Bourne Shell and *Bourne Shell Reference*

f

user information lookup program

See finger.

false

return exit status value non-zero

The **false** command returns an exit status value of nonzero (one). **False** and **true** are usually used in Bourne shell scripts that test for the appropriate status "false" or "true" before running (or failing to run) a list of commands.

---

false

EXAMPLE

```
while false
do
        command
            .
            .
            .

done
```

SEE ALSO

csh, sh, true

fg

execute job in foreground (C shell)

The fg command puts a job into the foreground and, if stopped, restarts it.

---

fg [*%job*]...

ARGUMENTS

%job specifies the number of the job to put into the foreground. The default is the current job.

EXAMPLE

The following command sequence:

```
% nroff myfile&
[1] 1234
% stop %1
% fg %1
nroff myfile
```

has the same effect as if you had originally typed:

```
% nroff myfile
```

SEE ALSO

bg, jobs, stop

fgrep

search file for pattern (fast)

The fgrep command searches files for lines matching a pattern. Normally, each line found is copied to the standard output. Fgrep patterns are fixed strings; it is fast and compact.

---

fgrep [*option...*] [*strings*] [*file*]

ARGUMENTS

-v displays all lines but those matching.

-x (exact) displays only lines matched in their entirety.

-c displays only a count of matching lines.

-l displays the names of files with matching lines (once) separated by newlines.

-n precedes each line by its relative line number in the file.

-b precedes each line by the block number on which it was found (sometimes useful in locating disk block numbers by context).

-i ignores the case of letters in making comparisons.

-s displays nothing except error messages (useful for checking the error status).

-f *file* the string list is taken from the specified *file*.

strings searches for lines that contain one of the specified *strings*, separated by newlines.

file specifies a file to search. The default is the standard input.

In all cases the file name is shown if there is more than one input file. Care should be taken when using the characters dollar sign, asterisk, left bracket, circumflex, vertical bar, parenthesis, and backslash in the *expression* as they are also meaningful to the shell. It is safest to enclose the entire *expression* argument in single quotes.

DIAGNOSTICS

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

NOTES

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

SEE ALSO

grep, egrep, ex, sed, sh

file

determine file type

The file command performs a series of tests in an attempt to classify files. If a file appears to be ASCII characters, *file* examines the first 512 bytes and tries to determine its language.

---

file *file...*

file –f *file*

ARGUMENTS

–f *file* specifies the name of a file containing a list of files to process.

NOTES

- File often makes mistakes. In particular it often suggests that shell scripts are C programs.
- File does not recognize *Pascal* or *Lisp*.
- On an FX/2800 Series machine, file reports that an executable ELF file has been stripped if this is the case.

find

search directory tree for specific files

The find command recursively descends the directory structure beginning at each specified pathname, seeking files that match a set of conditions. It can execute a command whenever it finds a matching file.

---

find *pathname ... expression*

find *pattern*

ARGUMENTS

pathname specifies a pathname.

expression is a boolean expression written in the primaries and operators described below.

pattern a filename pattern to look for in the pathname database.

DESCRIPTION

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

The second form of find rapidly searches the database /usr/lib/find/find.codes for all pathnames that match *pattern*. If escaped, normal shell globbing characters (* ? [ ]) may be used in *pattern*, but the matching differs in that no characters (e.g., /) have to be matched explicitly. As a special case, a simple pattern containing no globbing characters is matched as though it were *pattern*; if any globbing character appears there are no implicit globbing characters. The database is generated by executing the script /usr/lib/find/updatedb (the superuser must run this script) and contains the pathnames of all files that are publicly accessible. Usually the database is recomputed weekly when /usr/adm/weekly executes under the auspices of /usr/lib/crontab.

Primaries

In the following primaries, the argument *n* indicates a decimal integer where *+n* means more than *n*, *-n* means less than *n* and *n* means exactly *n*.

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

-cpio *file* writes the current file on the argument *file* in cpio format.

-exec *command*
executes the specified *command* and returns true if the command returns an exit status of zero. You can supply a pair of braces {} as an optional argument that is replaced by the current pathname. If you use the C shell, quote the braces (i.e., "{}"). Terminate the *command* by an escaped or quoted semicolon (i.e., \; or ";").

-fstype *type* returns true if the file is of type *type*. Valid types are 4.3, 4.2, nfs, tfs, and pc.

-group *gname* returns true if the file belongs to group *gname* (group name or numeric group ID).

-inum *n* returns true if the file has inode number *n*.

-links *n* returns true if the file has *n* links.

-ls always true; causes current pathname to be printed together with its associated statistics. These include (respectively) inode number, size in kilobytes (1024 bytes), protection mode, number of hard links, user, group, size in bytes, and modification time. If the file is a special file the size field will instead contain the major and minor device numbers. If the file is a symbolic link the pathname of the linked–to file is printed preceded by "->". The format is identical to that of "ls –gilds" (note however that formatting is done internally, without executing the *ls* program).

-mtime *n* returns true if the file has been modified in *n* days. **-mtime** *n* means modification time between *n* and *n*+1 days ago. **-mtime** 0 matches files modified in the last 24 hours. **-mtime** 1 matches files modified between 24 and 48 hours ago.

–mtime –3 means modification time between 0 and 3 days ago.

–mtime +3 means modification time between 3 days ago and infinite days ago.

-name *filename*
returns true if *filename* matches the name of an existing file. Normal shell argument syntax can be used if escaped or quoted (watch out for left bracket, question mark, and asterisk).

-newer *file* returns true if an existing file has been modified more recently than the argument *file*.

-nogroup returns true if the file belongs to a group not in the /etc/group database.

-nouser returns true if the file belongs to a user not in the /etc/passwd database.

-ok *command* same as **-exec** except that *find* asks for confirmation. The generated command is written on the standard output, then the standard input is read and the command executed only upon response y.

-perm *onum* returns true if the file permission flags exactly match the octal number *onum* (see chmod). If *onum* is prefixed by a minus sign, more flag bits (017777, see the system call *stat*) become significant and the flags are compared: *(flags&onum)==onum*.

-print always returns true; causes the current pathname to be displayed.

-prune always returns true; causes the search not to descend into the current pathname (if it is a directory).

-size *n* returns true if the file is *n* blocks long (512 bytes per block).

-type *c* returns true if the type of the file is *c*, where *c* is:

b	block special file
c	character special file
d	directory
f	plain file
l	symbolic link

-user *uname* returns true if the file belongs to the user *uname* (login name or numeric user ID).

-xdev Always true; causes find not to traverse down into a file system different from the one on which current argument pathname resides.

Operators

Primaries can be combined using the following operators (in order of decreasing precedence):

() indicate the grouping of primaries and operators. (Parentheses are significant to the shell and must be escaped.)

! indicates the negation of a primary (a unary *not* operator).

-o indicates the alternation of primaries (binary *or* operator). The absence of -o (juxtaposition) indicates the concatenation of primaries (binary *and* operator).

NOTES

- By default, find does not generate any output. To display the pathname of target files, use -print.
- More than one -newer option does not work properly.

EXAMPLE

To find all accessible files whose pathname contains 'find':

```
find find
```

This command removes all files in `/usr` named `a.out` or `*.o` that have not been accessed for a week:

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

FILES

/etc/passwd
/etc/group
/usr/lib/find/find.codes coded pathnames database

SEE ALSO

sh, test
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: fs

finger

user information lookup program

The finger (also f) command prints information about system users.

---

finger [*option...*] [*namelist...*]

ARGUMENTS

-b specifies a briefer version of the long output format.

-i specifies a quick list of users with idle time.

-f suppresses header for quick, narrow and short formats.

-m matches only on account name.

-l specifies the long output format (default with *name* argument).

-p suppresses printing of the `.plan` section of the long format.

-q specifies a quick list of one or more system users; shorter than the short format or the narrow format.

-s specifies the short output format (default with no *name* argument).

-w specifies the narrow output format; less information than the short format, but more than the quick format.

namelist is a list of one or more account names, first names, or last names of system users. When you enter more than one name, separate them with spaces.

With no *name* argument, finger displays information about each current system user.

DESCRIPTION

If you do not specify a *namelist*, the short output form is the default. If you specify a *namelist*, the long output form is the default.

The short output format includes:

- login name.
- full name.
- terminal name and write status (as an asterisk before the terminal name if write permission is denied).

- idle time (a single integer indicates minutes; a colon indicates hours and minutes; a d indicates days and hours).
- login time.
- office location (if known).
- phone number (if known).

The long (multi-line) output format includes all of the information in the short format as well as:

- home directory.
- login shell.
- plan (the contents of the file `.plan` in the user's home directory).
- project (the contents of the file `.project` in the user's home directory).

Finger may be used to look up users on a remote 4.3BSD machine. The format is to specify the users as "user@host". If the user name is left off, all the remote users are listed in the standard form.

NOTES

- The Concentrix-2800 command f is a synonym for finger and can be used in its place.
- Only the first line (70 characters) of the `.project` file is printed.
- The 4.3BSD version of finger filters non-printable characters (for example, escape sequences) out of the ~/.plan file.

FILES

/etc/utmp	who file
/etc/passwd	for users names, offices, ...
/usr/adm/lastlog	last login times
~/.plan	plans
~/.project	projects

SEE ALSO

passwd, w, who

fmt

simple text formatter (unsupported)

The fmt command invokes a simple text formatter that reads input files and produces on the standard output a version of its input with lines as close to 72 characters long as possible. The spacing at the beginning of the input lines is preserved in the output, as are blank lines and inter-word spacing.

---

fmt [*name...*]

ARGUMENTS

name specifies an input file. The default is the standard input.

DESCRIPTION

Fmt is meant to format mail messages prior to sending, but may also be useful for other simple tasks. For instance, within vi, the command

```
!}fmt
```

reformats a rpara, evening the lines.

NOTES

- The program was designed to be simple and fast. For more complex operations, the standard text processors are likely to be more appropriate.

SEE ALSO

nroff, mail

fold

fold long lines for narrow output device

The fold command is a filter that folds the contents of the specified files, breaking the lines to have a maximum width.

---

fold [*–width*] [*file...*]

ARGUMENTS

–width specifies the width of the lines in characters. *Width* should be a multiple of eight if tabs are present, otherwise the tabs should be expanded using *expand* first. The default *width* is 80.

file specifies an input file. The default is the standard input.

NOTES

- If underlining is present, it may be ruined by folding.

SEE ALSO

expand

for

iterated loop construct (Bourne shell)

The for command provides a simple iterated loop mechanism based on a list of string values. It is intended for use in shell scripts but is also useful for interactive work.

---

for *name*
in *wordlist*
do
  command
  command
  .
  .
done

ARGUMENTS

name is the name of a variable that acts as a loop index.

wordlist is a list of strings that are successively assigned to *name*.

command is any valid command.

DESCRIPTION

The loop iterates once for each value in the *wordlist*, assigning the values to the loop index variable *name*, which can be accessed by variable substitution. For example, the following shell script writes the letters "a", "b" and "c" to standard output, and prints the files named a, b and c.

```
for i
in a b c
do
  echo $i
  print $i
done
```

FOR LOOPS AT THE TERMINAL

The for command also can be used at the terminal when performing a number of similar commands. The shell prompts for input with a right angle bracket when reading the body of the loop. For example, to print the files named a, b and c, you could type:

```
$ for i
> in a b c
> do
> print $i
> done
$
```

NOTES

- The C shell foreach command is similar in function to for.

SEE ALSO

case, if, while, break, continue

foreach

iterated loop construct (C shell)

The foreach command provides a simple iterated loop mechanism based on a list of string values. It is intended for use in shell scripts but is also useful for interactive work.

---

foreach *name* (*wordlist*)
command
.
.
.
end

ARGUMENTS

name is the name of a variable that acts as a loop index.

wordlist is a list of strings that are successively assigned to *name*.

command is any valid command.

DESCRIPTION

The loop iterates once for each value in the *wordlist*, assigning the value to the loop index variable *name*, which can be accessed by variable substitution.

You can branch to the end of a foreach loop. The continue command causes the remaining commands on the line to execute, then transfers control to the end command, beginning another iteration. For example, consider the script `temp1`:

```
#
foreach i (a b c)
        echo $i
        if ($i == b) continue
        echo test failed
end

% temp1
a
test failed
b
c
test failed
```

You also can branch out of a foreach loop. The break command causes the remaining commands on the line to execute, then transfers control to the command after the end command, terminating the loop. For example, consider the script `temp2`:

```
#
foreach i (a b c)
        echo $i
        if ($i == b) break
        echo test failed
end
```

```
% temp2
a
test failed
b
```

FOREACH LOOPS AT THE TERMINAL

The foreach command can be used at the terminal when performing a number of similar commands. The shell prompts for input with a question mark when reading the body of the loop. For example, to search several user directories for the file `foobar`, you could type:

```
% find /usr/jones -name foobar -print
% find /usr/smith -name foobar -print
% find /usr/adams -name foobar -print
% find /usr/rogers -name foobar -print
```

These commands are very similar; thus, you can use *foreach* to do the same thing:

```
% foreach i (jones smith adams rogers)
? find /usr/$i -name foobar -print
? end
```

Very useful with loops are variables that contain lists of filenames or other words. For example:

```
% set list=(`ls`)
% foreach i ($list)
        .
        .
        .
```

The set command uses command substitution to assign the names of all the files in the current directory to the variable `list`. The foreach loop iterates over the value of `list` to perform any chosen function.

NOTES

- The Bourne shell for command is similar in function to foreach.

SEE ALSO

if, if-then-else, switch, while

fpr

convert Fortran carriage-control

The fpr command invokes a filter that transforms files formatted according to the Fortran carriage-control convention into files formatted according to UNIX line printer conventions.

---

fpr

DESCRIPTION

Fpr copies its input onto its output, replacing the carriage-control characters with characters that produce the intended effects when printed using lpr. The first character of each line determines the vertical spacing as follows:

Character	Vertical Space Before Printing
blank	One line
0	Two lines
1	To first line of next page
+	No advance

A blank line is treated as if its first character is a blank. A blank that appears as a carriage-control character is deleted. A zero is changed to a newline. A one is changed to a form feed. The effects of a plus sign are simulated with backspaces.

EXAMPLE

```
a.out | fpr | lpr

fpr < output | lpr
```

NOTES

- Results are undefined for input lines longer than 170 characters.

from

display mail header lines

The from command displays the mail header lines in your mailbox file to show who your mail is from.

---

from [**-s** *sender*] [*user*]

ARGUMENTS

-s *sender* displays only headers for mail sent by *sender*.

user examines the specified user's mailbox instead of your own.

FILES

/usr/spool/mail/*

SEE ALSO

biff, mail

fsplit

split up multi-routine Fortran file

The fsplit command attempts to split Fortran programs into separate routine files of the form *name.f*, where *name* is the name of the program unit (e.g. function, subroutine, block data, or program).

---

fsplit [**-e** *efile*]... [*file*]

ARGUMENTS

-e *efile* splits only the specified subprogram units into separate files. By default, each subprogram unit is split into a separate file.

file specifies an input file. The default is the standard input.

DESCRIPTION

The name for unnamed block data subprograms has the form *blkdtaNNN.f* where NNN is three digits and a file of this name does not already exist. For unnamed main programs the name has the form *mainNNN.f*. If there is an error in classifying a program unit, or if *name.f* already exists, the program unit is put in a file of the form *zzzNNN.f* where *zzzNNN.f* does not already exist.

EXAMPLE:

```
fsplit -e readit -e doit prog.f
```

splits readit and doit into separate files.

DIAGNOSTICS

- If names specified via the *-e* option are not found, a diagnostic is written to standard error.

NOTES

- Fsplit assumes the subprogram name is on the first noncomment line of the subprogram unit. Nonstandard source formats can confuse fsplit.
- fsplit demands that an END statement stand alone on a line, to be recognized.
- It is hard to use -e for unnamed main programs and block data subprograms because you must predict the created file name.
- If you are splitting FX/Fortran programs, watch out for things like:

```
INTEGER *2 FUNCTION( )
```

ftp

file transfer program

The ftp command is the user interface to the ARPANET standard File Transfer Protocol. The program allows you to transfer files to and from a remote network site.

---

ftp [-v] [-d] [-i] [-n] [-g] [*host*]

ARGUMENTS

Options can be specified at the command line, or to the command interpreter.

-v (verbose) forces ftp to show all responses from the remote server, as well as report on data transfer statistics.

-d enables debugging.

-i turns off interactive prompting during mutliple file transfers.

-n restrains ftp from attempting "auto-login" upon initial connection. If auto-login is enabled, ftp checks the `.netrc` file in the user's home directory for an entry describing an account on the remote machine, in the format:

`machine` *machinename* `login` *yourname* `password` *yourpassword*

replacing the italicized items as appropriate. (Be sure to `chmod 700 .netrc`.) If no entry exists, ftp uses the login name on the local machine as the user identity on the remote machine, and prompts for a password and, optionally, an account with which to login.

-g disables file name globbing.

DESCRIPTION

The client host with which ftp is to communicate can be specified on the command line. If this is done, ftp immediately attempts to establish a connection to an FTP server on that host; otherwise, ftp enters its command interpreter, displays the prompt `ftp>` and awaits instructions from the user. The following commands are recognized by ftp:

! Invoke a shell on the local machine.

$ *macro-name* [*args*]
Execute the macro *macro-name* that was defined with the macdef command. Arguments are passed to the macro unglobbed.

account [*passwd*]
Supply a supplemental password required by a remote system for access to re-

sources once a login has been successfully completed. If no argument is included, the user will be prompted for an account password in a non-echoing input mode.

append *local-file* [*remote-file*]
Append a local file to a file on the remote machine. If *remote-file* is left unspecified, the local file name is used in naming the remote file. File transfer uses the current settings for *type*, *format*, *mode*, and *structure*.

ascii Set the file transfer *type* to network ASCII. This is the default type.

bell Arrange that a bell be sounded after each file transfer command is completed.

binary Set the file transfer *type* to support binary image transfer.

bye Terminate the FTP session with the remote server and exit ftp.

case Toggle remote computer file name case mapping during *mget* commands. When case is on (default is off), remote computer file names with all letters in upper case are written in the local directory with the letters mapped to lower case.

cd *remote-directory*
Change the working directory on the remote machine to *remote-directory*.

cdup Change the remote machine working directory to the parent of the current remote machine working directory.

close Terminate the FTP session with the remote server, and return to the command interpreter.

cr Toggle carriage return stripping during ascii type file retrieval. Records are denoted by a carriage return/linefeed sequence during ascii type file transfer. When *cr* is on (the default), carriage returns are stripped from this sequence to conform with the UNIX single linefeed record delimiter. Records on non-UNIX remote systems may contain single linefeeds; when an ascii type transfer is made, these linefeeds may be distinguished from a record delimiter only when *cr* is off.

delete *remote-file*
Delete the file *remote-file* on the remote machine.

debug [*debug-value*]
Toggle debugging mode. If an optional *debug-value* is specified it is used to set the debugging level. When debugging is on, ftp displays each command sent to the remote machine, preceded by the string "-->".

dir [*remote-directory*] [*local-file*]
Display a listing of the directory contents in the directory, *remote-directory*, and, optionally, placing the output in *local-file*. If no directory is specified, the current working directory on the remote machine is used. If no local file is specified or the local file is -, output comes to the terminal.

disconnect
A synonym for close.

form *format*
Set the file transfer *form* to *format*. The default format is "file".

get *remote-file* [*local-file*]
Retrieve the *remote-file* and store it on the local machine. If the local file name is not specified, it is given the same name it has on the remote machine. The current settings for *type*, *form*, *mode*, and *structure are used while transferring the file*.

glob Toggle filename expansion for mdelete, mget and mput. If globbing is turned off with glob, the file name arguments are taken literally and not expanded. Globbing for mput is done as in csh. For mdelete and mget, each remote file name is expanded separately on the remote machine and the lists are not merged. Expansion of a directory name is likely to be different from expansion of the name of an ordinary file: the exact result depends on the foreign operating system and ftp server, and can be previewed by doing 'mls remote-files -'. Note: mget and mput are not meant to transfer entire directory subtrees of files. That can be done by transferring a tar archive of the subtree (in binary mode).

hash Toggle hash sign (#) printing for each data block transferred. The size of a data block is 1024 bytes.

help [*command*]
Display an informative message about the meaning of *command*. If no argument is given, ftp displays a list of the known commands.

lcd [*directory*]
Change the working directory on the local machine. If no *directory* is specified, the user's home directory is used.

ls [*remote-directory*] [*local-file*]
Display an abbreviated listing of the contents of a directory on the remote machine. If *remote-directory* is left unspecified, the current working directory is used. If no local file is specified, the output is sent to the terminal.

macdef *macro-name*
Define a macro. Subsequent lines are stored as the macro *macro-name*; a null line (consecutive newline characters in a file or carriage returns from the terminal) terminates macro input mode. There is a limit of 16 macros and 4096 total characters in all defined macros. Macros remain defined until a close command is executed. The macro processor interprets '$' and '\' as special characters. A '$' followed by a number (or numbers) is replaced by the corresponding argument on the macro invocation command line. A '$' followed by an 'i' signals that macro processor that the executing macro is to be looped. On the first pass '$i' is replaced by the first argument on the macro invocation command line, on the second pass it is replaced by the second argument, and so on. A '\' followed by any character is replaced by that character. Use the '\' to prevent special treatment of the '$'.

mdelete *remote-files*
Delete the specified files on the remote machine. If globbing is enabled, the specification of remote files is first expanded using ls.

mdir *remote-files local-file*
Obtain a directory listing of multiple files on the remote machine and place the result in *local-file*.

mget *remote-files*
Expand the *remote-files* on the remote machine and do a *get* for each file name thus produced. See *glob* for details on the filename expansion. Resulting file names will then be processed according to case, ntrans, and nmap settings. Files are transferred into the local working directory, which can be changed with 'lcd directory'; new local directories can be created with '! mkdir directory'.

mkdir *directory-name*
Make a directory on the remote machine.

mls *remote-files local-file*
Obtain an abbreviated listing of multiple files on the remote machine and place the result in *local-file*.

mode [*mode-name*]
Set the file transfer *mode* to *mode-name*. The default mode is "stream" mode.

mput *local-files*
Expand wild cards in the list of local files given as arguments and do a put for each file in the resulting list. See glob for details of filename expansion. Resulting file names will then be processed according to ntrans and nmap settings.

nmap [*inpattern outpattern*]
Set or unset the filename mapping mechanism. If no arguments are specified, the filename mapping mechanism is unset. If arguments are specified, remote filenames are mapped during mput commands and put commands issued without a specified remote target filename. If arguments are specified, local filenames are mapped during mget commands and get commands issued without a specified local target filename.

This command is useful when connecting to a non-UNIX remote computer with different file naming conventions or practices. The mapping follows the pattern set by *inpattern* and *outpattern*. *Inpattern* is a template for incoming filenames (which may have already been processed according to the ntrans and case settings). Variable templating is accomplished by including the sequences '$1', '$2', ..., '$9' in inpattern. Use '\' to prevent this special treatment of the '$' character. All other characters are treated literally, and are used to determine the nmap inpattern variable values.

For exmaple, given *inpattern* $1.$2 and the remote file name mydata.data, $1 would have the value "mydata", and $2 would have the value "data". The outpattern determines the resulting mapped filename. The sequences `$1`, `$2`, ..., `$9` are replaced by any value resulting from the *inpattern* template. The sequence $0 is replace by the original filename. Additionally, the sequence [seq1,seq2] is replaced by seq1 if seq1 is not a null string; otherwise it is replaced by seq2. For example, the command `nmap $1.$2.$3 [$1,$2].[$2,file]` yields the output filename myfile.data for input filenames `myfile.data` and `myfile.data.old`, myfile.file for the input filename `myfile`, and myfile.myfile for the input filename `myfile`. Spaces may be included in outpattern, as in the example: `nmap $1 |sed "s/ *$//" > $1` .

Use the '\' character to prevent special treatment of the '$', '[', ']', and ',' characters.

ntrans [*inchars* [*outchars*]]
Set or unset the filename character translation mechanism. If no arguments are specified, the filename character translation mechanism is unset. If arguments are specified, characters in remote filenames are translated during mput commands and put commands issued without a specified remote target filename. If arguments are specified, characters in local filenames are translated during mget commands and get commands issued without a specified local target filename.

This command is useful when connecting to a non-UNIX remote computer with different file naming conventions or practices. Characters in a filename matching a character in inchars are replaced with the corresponding character in outchars. If the character's position in inchars is longer than the length of outchars, the character is deleted from the file name.

open *host* [*port*]
Establish a connection to the specified *host* FTP server. An optional port number can be supplied, in which case, ftp attempts to contact an FTP server at that port. If the *auto-login* option is on (default), ftp also attempts to automatically log the user in to the FTP server (see below).

prompt
Toggle interactive prompting. Interactive prompting occurs during multiple file transfers to allow the user to selectively retrieve or store files. If prompting is turned off (default), any mget or mput transfers all files.

proxy *ftp-command*
Execute an ftp command on a secondary control connection. This command allows simultaneous connection to two remote ftp servers for transferring files between the two servers. The first proxy command should be an open, to establish the secondary control connection. Enter the command **proxy ?** to see other ftp commands executable on the secondary connection. The following commands behave differently when prefaced by proxy: open will not define new macros during the auto-login process, close will not erase existing macro definitions, get and mget transfer files from the host on the primary control connection to the host on the secondary control connection, and put, mput, and append transfer files from the host on the secondary control connection to the host on the primary control connection. Third party file transfers depend upon support of the ftp protocol PASV command by the server on the secondary control connection.

put *local-file* [*remote-file*]
Store a local file on the remote machine. If *remote-file* is left unspecified, the local file name is used in naming the remote file. File transfer uses the current settings for *type*, *format*, *mode*, and *structure*.

pwd Display the name of the current working directory on the remote machine.

quit A synonym for bye.

quote *arg1 arg2 ...*
The arguments specified are sent, verbatim, to the remote FTP server. A single FTP reply code is expected in return.

recv *remote-file* [*local-file*]
A synonym for get.

remotehelp [*command-name*]
Request help from the remote FTP server. If a *command-name* is specified it is supplied to the server as well.

rename [*from*] [*to*]
Rename the file *from* on the remote machine, to the file *to*.

reset Clear reply queue. This command re-synchronizes command/reply sequencing with the remote ftp server. Resynchronization may be neccesary following a violation of the ftp protocol by the remote server.

rmdir *directory-name*
Delete a directory on the remote machine.

runique
Toggle storing of files on the local system with unique filenames. If a file already exists with a name equal to the target local filename for a get or mget command, a .1 is appended to the name. If the resulting name matches another existing file, a .2 is appended to the original name. If this process continues up to .99, an error message is printed, and the transfer does not take place. The generated unique filename will be reported. Note that runique will not affect local files generated from a shell command (see below). The default value is off.

send *local-file* [*remote-file*]
A synonym for put.

sendport
Toggle the use of PORT commands. By default, ftp attempts to use a PORT command when establishing a connection for each data transfer. If the PORT command fails, ftp uses the default data port. When the use of PORT commands is disabled, no attempt is made to use PORT commands for each data transfer. This is useful for certain FTP implementations that do ignore PORT commands but, incorrectly, indicate they've been accepted.

status Show the current status of ftp.

struct [*struct-name*]
Set the file transfer *structure* to *struct-name*. By default "stream" structure is used.

sunique
Toggle storing of files on remote machine under unique file names. Remote ftp server must support ftp protocol STOU command for successful completion. The remote server will report unique name. Default value is off.

tenex Set the file transfer type to that needed to talk to TENEX machines.

trace Toggle packet tracing.

type [*type-name*]
Set the file transfer *type* to *type-name*. If no type is specified, the current type is displayed. The default type is network ASCII.

user *user-name* [*password*] [*account*]
Identify yourself to the remote FTP server. If the password is not specified and the server requires it, *ftp* prompts the user for it (after disabling local echo). If an account field is not specified, and the FTP server requires it, the user is prompted for it. Unless ftp is invoked with auto-login disabled, this process is done automatically on initial connection to the FTP server.

verbose
Toggle verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. In addition, if verbose is on, when a file transfer completes, statistics regarding the efficiency of the transfer are reported. By default, verbose is on.

? [*command*]
Is synonymous with help.

Command arguments that have embedded spaces can be quoted with quote marks.

ABORTING A FILE TRANSFER

To abort a file transfer, use the terminal interrupt key (usually CTRL/C). Sending transfers will be immediately halted. Receiving transfers will be halted by sending a ftp protocol ABOR command to the remote server, and discarding any further data received. The speed at which this is accomplished depends upon the remote server's support for ABOR processing. If the remote server does not support the ABOR command, an "ftp>" prompt will not appear until the remote server has completed sending the requested file.

The terminal interrupt key sequence will be ignored when ftp has completed any local processing and is awaiting a reply from the remote server. A long delay in this mode may result from the ABOR processing described above, or from unexpected behavior by the remote server, including violations of the ftp protocol. If the delay results from unexpected remote server behavior, the local ftp program must be killed by hand.

FILE NAMING CONVENTIONS

Files specified as arguments to ftp commands are processed according to the following rules.

1. If a hyphen is specified as a file name, the standard input (for reading) or standard output (for writing) is used.

2. If the first character of the file name is vertical bar, the remainder of the argument is interpreted as a shell command. Ftp then forks a shell, using popen *(Library Subroutines)* with the argument supplied, and reads (writes) from the stdout (stdin). If the shell command includes spaces, the argument must be quoted; for example: `"| ls -lt"`. A particularly useful example of this mechanism is: `dir |more`.

3. Failing the above checks, if globbing is enabled, local file names are expanded according to the rules used in the csh; c.f. the glob command.

4. For mget commands and get commands with unspecified local file names, the local filename is the remote filename, which may be altered by a case, ntrans, or nmap setting. The resulting filename may then be altered if runique is on.

5. For mput commands and put commands with unspecified remote file names, the remote filename is the local filename, which may be altered by a ntrans or nmap setting. The resulting filename may then be altered by the remote server if sunique is on.

FILE TRANSFER PARAMETERS

The FTP specification specifies many parameters that can affect a file transfer. The *type* can be one of `ascii`, `image` (binary), `ebcdic`, and `local byte size` (for Digital 10's and 20's mostly). Ftp supports the ASCII and image types of file transfer.

Ftp supports only the default values for the remaining file transfer parameters: *mode*, *form*, and *struct*.

THE .netrc FILE

The .netrc file contains login and initialization information used by the auto-login process. It resides in the user's home directory. The following tokens are recognized; they may be separated by spaces, tabs, or newlines:

machine *name*: Identify a remote machine name. The auto-login process searches the .netrc file for a machine token that matches the remote machine specified on the ftp command line or as an open command argument. Once a match is made, the subsequent .netrc tokens are processed, stopping when the end of file is reached or another machine token is encountered.

login *name*: Identify a user on the remote machine. If this token is present, the auto-login process will initiate a login using the specified name.

password *string*: Supply a password. If this token is present, the auto-login process will supply the specified string if the remote server requires a password as part of the login process. Note that if this token is present in the .netrc file, ftp aborts the auto-login process if the .netrc is readable by anyone besides the user.

account *string*: Supply an additional account password. If this token is present, the auto-login process will supply the specified string if the remote server requires an additional account password, or the auto-login process will initiate an ACCT command if it does not.

macdef *name*: Define a macro. This token functions like the ftp macdef command functions. A macro is defined with the specified name; its contents begin with the next .netrc line and continue until a null line (consecutive newline characters) is encountered. If a macro named init is defined, it is automatically executed as the last step in the auto-login process.

NOTES

- Correct execution of many commands depends upon proper behavior by the remote server.
- An error in the treatment of carriage returns in the 4.2BSD UNIX ascii-mode transfer code has been corrected. This correction may result in incorrect transfers of binary files to and from 4.2BSD servers using the ascii type. Avoid this problem by using the binary image type.

get

get a version of an SCCS file

Get generates an ASCII text file from each named SCCS file according to the specifications given by its keyletter arguments, which begin with -. The generated text is normally written into a file called the *g-file* whose name is derived from the SCCS file name by simply removing the leading s.; (see FILES below).

---

get [*option...*] *file...*

ARGUMENTS

The arguments can be specified in any order, but all keyletter arguments apply to all named SCCS files. Each keyletter argument is explained as though only one SCCS file is to be processed, but the effects of any keyletter argument apply independently to each named file.

-r*SID* specifies the *S*CCS *ID*entification string (SID) of the version (delta) of an SCCS file to be retrieved. The table below shows, for the most useful cases, what version of an SCCS file is retrieved (as well as the SID of the version to be eventually created by delta if the -e keyletter is also used), as a function of the SID specified.

-c*cutoff* specifies the *cutoff* date-time, in the form:

YY[*MM*[*DD*[*HH*[*MM*[*SS*]]]]]

No changes (deltas) to the SCCS file that were created after the specified *cutoff* date-time are included in the generated ASCII text file. Units omitted from the date-time default to their maximum possible values; that is, -c7502 is equivalent to -c750228235959. Any number of non-numeric characters can separate the various two-digit pieces of the *cutoff* date-time. This feature allows one to specify a *cutoff* date in the form: "-c77/2/2 9:22:25".

-e indicates that the get is for the purpose of editing or making a change (delta) to the SCCS file via a subsequent use of *delta*. The -e keyletter used in a get for a particular version (SID) of the SCCS file prevents further *get*s for editing on the same SID until delta is executed or the j (joint edit) flag is set in the SCCS file (see *admin*). Concurrent use of get -e for different SIDs is always allowed.

If the *g-file* generated by get with an -e keyletter is accidentally ruined in the process of editing it, it can be regenerated by reexecuting the get command with the -k keyletter in place of the -e keyletter.

SCCS file protection specified via the ceiling, floor, and authorized user list stored in the SCCS file (see admin) are enforced when the -e keyletter is used.

-b is used with the **-e** keyletter to indicate that the new delta should have an SID in a new branch as shown in the table below. This keyletter is ignored if the **b** flag is not present in the file (see **admin**) or if the retrieved *delta* is not a leaf *delta*. (A leaf *delta* is one that has no successors on the SCCS file tree.) Note: A branch *delta* can always be created from a non-leaf *delta*.

-i*list* specifies a *list* of deltas to be included (forced to be applied) in the creation of the generated file. The *list* has the following syntax:

<list> ::= <range> | <list> , <range>
<range> ::= SID | SID - SID

SID, the SCCS Identification of a delta, can be in any form shown in the "SID Specified" column of the table below. Partial SIDs are interpreted as shown in the "SID Retrieved" column of the table below.

-x*list* specifies a *list* of deltas to be excluded (forced not to be applied) in the creation of the generated file. See the **-i** keyletter for the *list* format.

-k suppresses replacement of identification keywords (see below) in the retrieved text by their value. The **-k** keyletter is implied by the **-e** keyletter.

-l[p] causes a delta summary to be written into an *l-file*. If **-lp** is used, an *l-file* is not created; the delta summary is written on the standard output instead. See *FILES* for the format of the *l-file*.

-p causes the text retrieved from the SCCS file to be written on the standard output. No *g-file* is created. All output that normally goes to the standard output goes to file descriptor 2 instead, unless the **-s** keyletter is used, in which case it disappears.

-s suppresses all output normally written on the standard output. However, fatal error messages (which always go to file descriptor 2) remain unaffected.

-m causes each text line retrieved from the SCCS file to be preceded by the SID of the delta that inserted the text line in the SCCS file. The format is: SID, followed by a horizontal tab, followed by the text line.

-n causes each generated text line to be preceded with the %M% identification keyword value (see below). The format is: %M% value, followed by a horizontal tab, followed by the text line. When both the **-m** and **-n** keyletters are used, the format is: %M% value, followed by a horizontal tab, followed by the **-m** keyletter generated format.

-g suppresses the actual retrieval of text from the SCCS file. It is primarily used to generate an *l-file*, or to verify the existence of a particular SID.

-t accesses the most recently created ("top") delta in a given release (e.g., **-r1**), or release and level (e.g., **-r1.2**).

-a*seq-no.* specifies the delta sequence number of the SCCS file delta (version) to be retrieved (see the file format *sccsfile*). This keyletter is used by the **comb**

command; it is not a generally useful keyletter, and users should not use it. If both the -r and -a keyletters are specified, the -a keyletter is used. Care should be taken when using the -a keyletter in conjunction with the -e keyletter, as the SID of the delta to be created may not be what one expects. The -r keyletter can be used with the -a and -e keyletters to control the naming of the SID of the delta to be created.

file specifies a file to be processed. If a directory is named, *get* behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored. If a name of "-" is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file to be processed. Again, non-SCCS files and unreadable files are silently ignored.

DESCRIPTION

For each file processed, get responds (on the standard output) with the SID being accessed and with the number of lines retrieved from the SCCS file.

If the -e keyletter is used, the SID of the delta to be made appears after the SID accessed and before the number of lines generated. If there is more than one named file or if a directory or standard input is named, each file name is printed (preceded by a new-line) before it is processed. If the -i keyletter is used included deltas are listed following the notation "Included"; if the -x keyletter is used, excluded deltas are listed following the notation "Excluded".

Table 1 Determination of SCCS Identification String

SID* Specified	-b Keyletter Used**	Other Conditions	SID Retrieved	SID of Delta to be Created
none#	no	R defaults to mR	mR.mL	mR.(mL+1)
none#	yes	R defaults to mR	mR.mL	mR.mL.(mB+1).1
R	no	R > mR	mR.mL	R.1***
R	no	R = mR	mR.mL	mR.(mL+1)
R	yes	R > mR	mR.mL	mR.mL.(mB+1).1
R	yes	R = mR	mR.mL	mR.mL.(mB+1).1
R	-	R < mR and R does *not* exist	hR.mL##	hR.mL.(mB+1).1
R	-	Trunk successor in release > R and R exists	R.mL	R.mL.(mB+1).1
R.L	no	No trunk successor	R.L	R.(L+1)
R.L	yes	No trunk successor	R.L	R.L.(mB+1).1
R.L	-	Trunk successor in release > R	R.L	R.L.(mB+1).1
R.L.B	no	No branch successor	R.L.B.mS	R.L.B.(mS+1)
R.L.B	yes	No branch successor	R.L.B.mS	R.L.(mB+1).1
R.L.B.S	no	No branch successor	R.L.B.S	R.L.B.(S+1)
R.L.B.S	yes	No branch successor	R.L.B.S	R.L.(mB+1).1
R.L.B.S	-	Branch successor	R.L.B.S	R.L.(mB+1).1

* "R", "L", "B", and "S" are the "release", "level", "branch", and "sequence" components of the SID, respectively; "m" means "maximum". Thus, for example, "R.mL" means "the maximum level number within release R"; "R.L.(mB+1).1" means "the first sequence number on the *new* branch (i.e., maximum branch number plus one) of level L within release R". Note that if the SID specified is of the form "R.L", "R.L.B", or "R.L.B.S", each of the specified components *must* exist.

** The -b keyletter is effective only if the b flag (see admin) is present in the file. An entry of "-" means "irrelevant".

*** This is used to force creation of the *first* delta in a *new* release.

\# This case applies if the d (default SID) flag is *not* present in the file. If the d flag *is* present in the file, the SID obtained from the d flag is interpreted as if it had

been specified on the command line. Thus, one of the other cases in this table applies.

"hR" is the highest *existing* release that is lower than the specified, *nonexistent*, release R.

IDENTIFICATION KEYWORDS

Identifying information is inserted into the text retrieved from the SCCS file by replacing *identification keywords* with their value wherever they occur. The following keywords can be used in the text stored in an SCCS file:

Keyword *Value*

%M% Module name: either the value of the m flag in the file (see *admin*), or if absent, the name of the SCCS file with the leading s. removed.

%I% SCCS identification (SID) (%R%.%L%.%B%.%S%) of the retrieved text.

%R% Release.

%L% Level.

%B% Branch.

%S% Sequence.

%D% Current date (YY/MM/DD).

%H% Current date (MM/DD/YY).

%T% Current time (HH:MM:SS).

%E% Date newest applied delta was created (YY/MM/DD).

%G% Date newest applied delta was created (MM/DD/YY).

%U% Time newest applied delta was created (HH:MM:SS).

%Y% Module type: value of the t flag in the SCCS file (see *admin*).

%F% SCCS file name.

%P% Fully qualified SCCS file name.

%Q% The value of the q flag in the file (see *admin*).

%C% Current line number. This keyword is intended for identifying messages output by the program such as "this shouldn't have happened" type errors. It is *not* intended to be used on every line to provide sequence numbers.

%Z% The four-character string @(#) recognizable by *what*.

%W% A shorthand notation for constructing *what* strings for the UNIX System program files. %W% = %Z%%M%<horizontal-tab>%I%

%A% Another shorthand notation for constructing *what* strings for non-UNIX System program files. %A% = %Z%%Y% %M% %I%%Z%

FILES

Several auxiliary files can be created by get, These files are known generically as the *g–file*, *l–file*, *p–file*, and *z–file*. The letter before the hyphen is called the tag. An auxiliary file name is formed from the SCCS file name: the last component of all SCCS file names must be of the form s.*module–name*, the auxiliary files are named by replacing the leading s with the tag. The *g–file* is an exception to this scheme: the *g–file* is named by removing the s. prefix. For example, s.xyz.c, the auxiliary file names would be xyz.c, l.xyz.c, p.xyz.c, and z.xyz.c, respectively.

The *g–file*, which contains the generated text, is created in the current directory (unless the –p keyletter is used). A *g–file* is created in all cases, whether or not any lines of text were generated by the *get*. It is owned by the real user. If the –k keyletter is used or implied its mode is 644; otherwise its mode is 444. Only the real user need have write permission in the current directory.

The *l–file* contains a table showing which deltas were applied in generating the retrieved text. The *l–file* is created in the current directory if the –l keyletter is used; its mode is 444 and it is owned by the real user. Only the real user need have write permission in the current directory.

Lines in the *l–file* have the following format:

A blank character if the delta was applied; * otherwise.

A blank character if the delta was applied or wasn't applied and ignored; * if the delta wasn't applied and wasn't ignored.

A code indicating a "special" reason why the delta was or was not applied:

"I": Included.
"X": Excluded.
"C": Cut off (by a –c keyletter).

Blank.

SCCS identification (SID).

Tab character.

Date and time (in the form YY/MM/DD HH:MM:SS) of creation.

Blank.

Login name of person who created *delta*.

The comments and MR data follow on subsequent lines, indented one horizontal tab character. A blank line terminates each entry.

The *p–file* is used to pass information resulting from a get with an –e keyletter along to *delta*. Its contents are also used to prevent a subsequent execution of get with an –e keyletter for the same SID until *delta* is executed or the joint edit flag, j, (see admin) is set in the SCCS file. The *p–file* is created in the directory containing the SCCS file and the effective user must have write permission in that directory. Its mode is 644 and it is owned by the effective user. The format of the *p–file* is: the gotten SID, followed by a blank, followed by the SID that the new delta will have when it is made, followed by a blank, followed by the login name of the real user, followed by a blank, followed by the date-time the get was executed, followed by a blank and the –i keyletter argument if it was present, followed by a blank and the –x keyletter argument if it was present, followed by a new-line. There can be an arbitrary number of lines in the *p–file* at any time; no two lines can have the same new delta SID.

The *z–file* serves as a *lock-out* mechanism against simultaneous updates. Its contents are the binary (two bytes) process ID of the command (i.e., get) that created it. The *z–file* is created in the directory containing the SCCS file for the duration of get. The same protection restrictions as those for the *p–file* apply for the *z–file*. The *z–file* is created mode 444.

DIAGNOSTICS

- Use help for explanations.

NOTES

- If the effective user has write permission (either explicitly or implicitly) in the directory containing the SCCS files, but the real user doesn't, then only one file can be named when the –e keyletter is used.

SEE ALSO

sccs, admin, delta, help, prs, what
Concentrix-2800 Programming Tools Guide: Source Code Control System – System V Interface
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: sccsfile

glob

echo arguments delimited by nulls (C shell)

The glob command displays its arguments on the standard output (like *echo*) but does not recognize backslash escapes. The words are delimited by null characters in the output. The glob command is useful for programs that wish to use the shell to filename expand a list of words.

---

glob [*arg...*]

ARGUMENTS

arg can be a quoted string, an ambiguous file reference, a shell variable, and so forth.

SEE ALSO

echo

goto

unconditional branch control structure (C shell)

The goto command branches to the specified label.

---

name:
    command
    .
    .
    .
    goto *name*

ARGUMENTS

name is filename and command expanded to yield a string of the form `label`.

DESCRIPTION

The shell rewinds its input as much as possible and searches for a line of the form label: possibly preceded by blanks or tabs. Execution continues after the specified line.

SEE ALSO

foreach, if, if-then-else, switch, while

gprof

display call graph profile data

The gprof command produces an execution profile of a program.

---

gprof [*option...*] [*objfile* [*profile...*]]

ARGUMENTS

-a suppresses the display of statically declared functions. If this option is given, all relevant information about the static function (*e.g.*, time samples, calls to other functions, calls from other functions) belongs to the function loaded just before the static function in the a.out file.

-A processes symbol names, produced by the Ada compiler, containing embedded periods (.). These names normally are considered file names and are excluded from the symbol table.

-b suppresses the display of a description of each field in the profile.

-c the static call graph of the program is discovered by a heuristic that examines the text space of the object file. Static-only parents or children are indicated with call counts of 0.

-e *name* suppresses the display of the graph profile entry for routine *name* and all its descendants (unless they have other ancestors that aren't suppressed). More than one -e option can be given. Only one *name* can be given with each -e option.

-E *name* suppresses the display of the graph profile entry for routine *name* (and its descendants) as -e, above, and also excludes the time spent in *name* (and its descendants) from the total and percentage time computations. (For example, -E mcount -E mcleanup is the default.)

-f *name* displays the graph profile entry of only the specified routine *name* and its descendants. More than one -f option can be given. Only one *name* can be given with each -f option.

-F *name* displays the graph profile entry of only the routine *name* and its descendants (as -f, above) and also uses only the times of the displayed routines in total time and percentage computations. More than one -F option can be given. Only one *name* can be given with each -F option. The -F option overrides the -E option.

-s a profile file gmon.sum is produced that represents the sum of the profile information in all the specified profile files. This summary profile file can be given to subsequent executions of gprof (probably also with a -s) to accumulate profile data across several runs of an a.out file.

-z displays routines that have zero usage (as indicated by call counts and accumulated time), useful in conjunction with the -c option for discovering which routines were never called.

objfile specifies an object file (default a.out).

profile specifies a call graph profile file (default gmon.out).

DESCRIPTION

The effect of called routines is incorporated in the profile of each caller. The profile data is taken from the call graph profile file (gmon.out default) that is created by programs that are compiled with the -pg option of cc and pc. That option also links in versions of the library routines that are compiled for profiling.

The symbol table in the named object file (a.out default) is read and correlated with the call graph profile file. If more than one profile file is specified, the gprof output shows the sum of the profile information in the given profile files.

First, a flat profile is given, similar to that provided by prof. This listing gives the total execution times and call counts for each of the functions in the program, sorted by decreasing time.

Next, these times are propagated along the edges of the call graph. Cycles are discovered, and calls into a cycle are made to share the time of the cycle. A second listing shows the functions sorted according to the time they represent including the time of their call graph descendents. Below each function entry is shown its (direct) call graph children, and how their times are propagated to this function. A similar display above the function shows how this function's time and the time of its descendents is propagated to its (direct) call graph parents.

Cycles are also shown, with an entry for the cycle as a whole and a listing of the members of the cycle and their contributions to the time and call counts of the cycle.

FILES

a.out	the namelist and text space.
gmon.out	dynamic call graph and profile.
gmon.sum	summarized dynamic call graph and profile.

NOTES

- Beware of quantization errors. The granularity of the sampling is shown, but remains statistical at best. We assume that the time for each execution of a function can be expressed by the total time for the function divided by the number of times the function is called. Thus the time propagated along the call graph arcs to parents of that function is directly proportional to the number of times that arc is traversed.

- Parents that are not themselves profiled have the time of their profiled children propagated to them, but they appear to be spontaneously invoked in the call graph listing, and do not have their time propagated further. Similarly, signal

catchers, even though profiled, appear to be spontaneous (although for more obscure reasons). Any profiled children of signal catchers should have their times propagated properly, unless the signal catcher was invoked during the execution of the profiling routine, in which case all is lost.

- The profiled program must call exit *(System Calls)* or return normally for the profiling information to be saved in the gmon.out file.

SEE ALSO

pcc, lprof, scc
Concentrix-2800 System Reference: System Calls and C Library Subroutines: monitor, profil
Concentrix-2800 Programming Tools Guide: Gprof: A Call Graph Execution Profiler

graph

draw a graph

The graph command takes pairs of numbers from the standard input as abscissas and ordinates of a graph. Successive points are connected by straight lines. The graph is encoded on the standard output for display by the plot filters.

---

graph [*option...*]

ARGUMENTS

-a [*spacing* [*start*]]
supplies abscissas automatically; *x* specifies the spacing (default 1); *y* specifies the starting point (default 0 or lower limit given by the -x option).

-b breaks (disconnects) the graph after each label in the input.

-c *string* specifies a default label for each point.

-D uses spacing for double size characters.

-g *style* specifies grid style:

0	no grid
1	frame with ticks
2	full grid (default)
3	same as 1 with scale numbes drawn next to tick marks
4	same as 2 with scale numbers drawn next to tick marks

-l *string* specifies a label for the graph.

-m *mode* specifies the mode (style) of connecting lines: 0 disconnected, 1 connected (default). Some devices give distinguishable line styles for other small integers.

-s saves the screen, does not erase before plotting. A legend indicating grid range is produced with a grid unless the -s option is present.

-x [l] [*lowerx* [*upperx* [*spacing*]]]
1 specifies that the *x* axis is logarithmic. The next one or two arguments specify lower (and upper) *x* limits. The third argument specifies grid spacing on the *x* axis. Normally these quantities are determined automatically. If a specified lower limit exceeds the upper limit, the axis is reversed.

-y [l] [*lowery* [*uppery* [*spacing*]]]
1 specifies that the *y* axis is logarithmic. The next one or two arguments specify lower (and upper) *y* limits. The third argument specifies grid spacing on the *y* axis. Normally these quantities are determined automatically.If a specified lower limit exceeds the upper limit, the axis is reversed.

-h *n* specifies the fraction of space for height.

-w *n* specifies the fraction of space for width.

-r *n* specifies the fraction of space to move right before plotting.

-u *n* specifies the fraction of space to move up before plotting.

-t transposes the horizontal and vertical axes. (Option -x now applies to the vertical axis.)

-X *string* specifies a label for the x-axis.

-Y *string* specifies a label for the y-axis.

DESCRIPTION

If the coordinates of a point are followed by a nonnumeric string, that string is displayed as a label beginning on the point. Labels can be surrounded with quotes "...", in which case they can be empty or contain blanks and numbers; labels never contain newlines.

NOTES

- Graph stores all points internally and drops those for which there isn't room.
- Segments that run out of bounds are dropped, not windowed.
- Logarithmic axes cannot be reversed.

SEE ALSO

spline, plot

grep

search a file for a pattern

The grep command searches files for lines matching a pattern. Normally, each line found is copied to the standard output. Grep patterns are limited regular expressions in the style of ex; it uses a compact nondeterministic algorithm.

---

grep [*option...*] *expression* [*file...*]

ARGUMENTS

-a displays the matching line and all lines that follow.

-b precedes each line by the block number on which it was found (sometimes useful in locating disk block numbers by context).

-c displays only a count of matching lines.

-d*n* displays the matching line, the *n* preceding lines, and the *n* following lines; *n* defaults to 1.

-e *expression*
is the same as an *expression* argument, but useful when the *expression* begins with a hyphen.

-i ignores the case of letters in making comparisons.

-j*n* parallelizes the search process. *n* specifies the number of parallel groups. *n* defaults to 1, and can be as high as the number of (S)CEs plus one.

-l displays the names of files with matching lines (once) separated by newlines.

-n precedes each line by its relative line number in the file.

-s displays nothing except error messages (useful for checking the error status).

-v displays all lines but those matching.

-w searches for the expression as a word (as if surrounded by '\<' and '\>' (see *ex*)).

expression is a limited regular expression in the style of *ex*.

file specifies a file to search. The default is the standard input.

In all cases the file name is shown if there is more than one input file. Care should be taken when using the characters dollar sign, asterisk, left bracket, circumflex, vertical bar, parenthesis, and backslash in the *expression* as they are also meaningful to the Shell. It is safest to enclose the entire *expression* argument in single quotes.

DIAGNOSTICS

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

NOTES

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

SEE ALSO

egrep, fgrep, ex, sed, sh

groups

show group memberships

The groups command shows the groups to which you or a specified user belong. Each system user belongs to a group specified in the password file `/etc/passwd` and possibly to other groups as specified in the file `/etc/group`.

---

groups [*user*]

DESCRIPTION

When a new file is created it is given the group of the containing directory. Any user belonging to that group has group access to the file.

FILES

/etc/passwd, /etc/group

SEE ALSO

Concentrix-2800 System Reference: System Calls and C Library Subroutines: setgroups

hashstat

print hash table statistics (C shell)

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

---

hashstat

SEE ALSO

rehash, unhash

head

display first few lines of a file

The head command invokes a filter that displays the first few lines of each of the specified files.

---

head [*–count*] [*file...*]

ARGUMENTS

–count specifies the number of lines to output (default is ten).

file specifies an input file.

SEE ALSO

tail

help

describe on-line documentation

If no arguments are specified, the help command performs a limited set of on-line documentation functions in interactive mode. If arguments are specified, the *help* command is the same as the *man* command. For help on the man command, type man man (or help man).

In reading a man page, press SPACE to read the next page or RETURN to read a line at a time (unless the - option is in effect). Type q to quit.

---

help [*same options as the man command*]

DESCRIPTION

The Concentrix-2800 on-line documentation consists of man pages. Each man page contains the following information:

- **Name** – Topic name. Typically, but not necessarily, the name of a command, routine, or other system component. The name of this man page is help.
- **Whatis line** – A one-line description of the topic. The whatis line for this man page is describe on-line documentation.
- **Content** – A brief description, a synopsis of the command or routine syntax, option descriptions, a full description, cross-references to files and other man pages, and other information, as necessary.

Each command is in a section denoted by a digit or letter, and optionally a subsection denoted by a letter. The names and whatis lines are duplicated in a separate entity called the whatis data base.

The following commands access the man and whatis data bases:

- **man** – Displays information about a specified topic. The topic name (typically the name of a command or routine) must be exact. The following example displays information on the cp command:

  ```
  man cp
  ```

  The next example displays information on the kill command in section 1, subsection b:

  ```
  man 1b kill
  ```

- **whatisin** – Displays the sections of the man database (no options); or displays the names and whatis lines for all topics in specified sections. The following example displays what is in section 1:

```
whatisin 1
```

- **whatis** – Displays the names and whatis lines for all topics in all sections with the specified topic names. (See also man –f and man –a.) The following examples display what kill topics exist:

```
whatis kill
```

- **apropos** – Displays the names and whatis lines for all topics whose names or whatis lines contain a specified character string. (See also man –k.) The following example displays every topic whose name or whatis line contains the word copy:

```
apropos copy
```

The head of the man data base is the directory /usr/man. Subdirectories named man0, man1, man2, and so on, contain the man pages. The name of each man page is the topic name, a period, the section, and (if applicable) the subsection. For example, the file containing the man page for the cp topic in section 1 is named /usr/man/man1/cp.1; the files containing the man pages for the topic kill in section 1, subsections 1b and 1c, are named /usr/man/man1/kill.1b and /usr/man/man1/kill.1c.

The man pages are in nroff source format and must be processed (formatted) before being displayed, which causes some delay. A man data base may also contain formatted files ready for immediate display. These files have identical names to the man files but reside in the subdirectories cat0, cat1, cat2, and so on.

The whatis data base is the single file /usr/man/whatis, and is used by the whatisin, whatis, and apropos commands. The whatisin command also uses the file /usr/man/whatisin.

NOTES

To access SCCS help, type /usr/sys5/help, or place /usr/sys5 ahead of /usr/alliant in your path.

FILES

/usr/man/man?/*	on-line documentation in nroff source format
/usr/man/cat?/*	on-line documentation in nroff output format
/usr/man/whatis	whatis data base
/usr/man/whatisin.top	sections of man data base

SEE ALSO

apropos, man, whatis, whatisin

helpsccs

get help on SCCS commands and errors

The help command finds information to explain a message from an SCCS command or explain the use of an SCCS command. The response of the program is the explanatory information related to the argument, if there is any. When all else fails, try `help stuck`.

/usr/sys5/help [*arg...*]

ARGUMENTS

Zero or more arguments can be supplied. If no arguments are given, *help* prompts for one. Arguments can be either message numbers (which normally appear in parentheses following messages) or command names, of one of the following types:

- Begins with non-numerics, ends in numerics. The non-numeric prefix is usually an abbreviation for the program or set of routines that produced the message (e.g., `ge6`, for message 6 from the *get* command).
- Does not contain numerics (as a command, such as `get`).
- Is all numeric (e.g., `212`).

FILES

/usr/lib/help	directory containing files of message text.
/usr/lib/help/helploc	file containing locations of help files not in /usr/lib/help.

DIAGNOSTICS

- Use help for explanations.

SEE ALSO

sccs, admin, delta, get, prs, what
Concentrix-2800 Programming Tools Guide: Source Code Control System – System V Interface
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: sccsfile

history

display most recent commands (C shell)

The history command displays a numbered list of the most recent commands you have typed.

---

history [-r] [-h] [*n*]

ARGUMENTS

-r reverses the order of the commands so that the most recent appears first. By default, the oldest command appears first.

-h suppresses the leading numbers (useful as input to source -h).

n displays the *n* most recent commands.

SEE ALSO

bg, fg
Concentrix-2800 System User's Guide: Using the C shell: Variables (histchars, history, savehist)

hostid

set/display host system identifier

The hostid command displays the identifier of the current host in hexadecimal. A host identifier is expected to be unique across all hosts and is normally set to the host's Internet address.

---

hostid [*identifier*]

ARGUMENTS

identifier is a hexadecimal number that specifies a host identifier.

DESCRIPTION

If the argument is omitted, the host identifier is displayed.

If the argument is included, the host identifier is set. The host identifier may be set only by the superuser. This is usually done in the startup script /etc/rc.local.

SEE ALSO

Concentrix-2800 System Reference: System Calls and C Library Subroutines: gethostid, sethostid

hostname

set/display host system name

The hostname command displays the name of the current host, as given before the `login` prompt.

---

hostname [*name*]

ARGUMENTS

name specifies a host name (super-user only). This is usually done in the startup script /etc/rc.local.

SEE ALSO

Concentrix-2800 System Reference: System Calls and C Library Subroutines: gethostname, sethostname

if

simple two-way branch control structure (C shell)

The if command provides a simple, two-way branching mechanism that is intended for use in shell scripts.

---

if (*expression*) *command*

ARGUMENTS

expression is an numeric, filename, or command expression as described in the *Concentrix-2800 System User's Guide, C Shell Reference*. If *expression* returns true (non-zero), the *command* is executed.

command cannot be: a pipeline, a background command, commands separated by semicolons, or another control command.

NOTES

You also can write the *if* command this way:

```
if (expression) \
        command
```

The backslash must *immediately* precede the end-of-line.

SEE ALSO

foreach, if-then-else, switch, while

if–then–else

multiple two-way branch control structure (C shell)

The if-then-else command provides a multi-way branching mechanism that is intended for use in shell scripts.

---

if (*expression*) **then**
    command
    .
    .
    .
else if (*expression*) **then**
    command
    .
    .
    .
else
    command
    .
    .
    .
endif

ARGUMENTS

expression is an numeric, filename, or command expression as described in the *Concentrix-2800 System User's Guide, C Shell Reference*. The shell evaluates *expressions* until it finds one that returns true (non-zero) and executes its block of *commands*. If control reaches the `else` clause, its block of *commands* are executed.

command is any valid command.

NOTES

- The `else if` clauses and the `else` clause are optional; you can have several `else if` clauses but only one `else` clause.
- The placement of the keywords in an if-then-else command is *not* flexible. The following two formats are *not* valid:

```
if (expression)
then
        command
            .
            .
            .
endif
```

and

```
if (expression) then command endif
```

- The Bourne shell if command is similar in function to if-then-else.

SEE ALSO

foreach, if, switch, while

if

multiple two-way branch control structure (Bourne shell)

The Bourne shell if command provides a multi-way branching mechanism that is intended for use in Bourne shell scripts. The Bourne shell if command is similar to the C-shell if-then-else command.

---

if *list*
  then *list*
elif *list*
  then *list*
else
  list
fi

ARGUMENTS

list Depending on where it occurs in the syntax of the if command, *list* is either executed and its value tested, or it is just executed. See below for a full description.

DESCRIPTION

The *list* following the **if** keyword is executed, and its value examined. If the value is true (zero), the *list* following the first **then** keyword is executed. After the *list* following the **then** is executed, control passes to the statement following the **fi** keyword.

If the value of the *list* following the **if** keyword is false (non-zero), the *list* following the first **elif** is executed and its value examined. If this value is true, the *list* following the matching **then** is executed. After this *list* is executed, control passes to the statement following the **fi** keyword.

If both the *lists* following the **if** and the first **elif** are false, each succeeding *list* following an **elif** is executed and its result evaluated. When one of these *lists* is true, the *list* following the matching **then** is executed as well. When a *list* following a **then** is executed, control passes next to the statement following the **fi** keyword.

If no *lists* following the **if** or an **elif** are true, the *list* following the **else** keyword is executed.

NOTES

- Each *list* following the **if** or an **elif** is executed before its value is examined. Any action that these *lists* take will occur, therefore, even if the *list* evaluated to false.

- An **if** statement must contain the **if** and **fi** keywords, but need not contain any **elif** or **else** clauses.

- The placement of the keywords in an *if* command is not flexible. The following format is not valid:

```
if list then
  list
elif list then
  list
else
  list
fi
```

EXAMPLE

The Bourne shell if command is shown in the following example. Suppose that the name of this shell script is **talk**. Typing "**talk hello**" causes the Bourne shell to respond "**Hello to you too.**" Typing "**talk goodbye**" causes the Bourne shell to respond "**See you later.**" Typing "**talk foo**" causes the Bourne shell to respond "**Nice to talk to you.**" followed by "**See you later.**"

```
if test $1 = hello
  then echo Hello to you too.
elif test $1 = goodbye
  then echo See you later.
else
  echo Nice to talk to you. ; echo See you later.
fi
```

indent

indent and format C program source

The indent command invokes a C program formatter. Specifically, indent:

- indents code lines.
- aligns comments.
- inserts spaces around operators where necessary.
- breaks up declaration lists such as, `int a,b,c;`.

---

indent *input* [*output*] [*option...*]

ARGUMENTS

Options can appear before or after the file names.

input specifies an input file. If you only specify an input file, the formatting is done in-place. The formatted file is written back into *input* and a backup copy of the input file is written in the current directory. If *input* is named **file**, the backup file is named **file.BAK**.

output specifies an output file. Indent checks to make sure *output* is different from *input*.

-bad,-nbad If -bad is specified, a blank line is forced after every block of declarations. Default: -nbad.

-bap,-nbap If -bap is specified, a blank line is forced after every procedure body. Default: -nbap.

-bbb,-nbbb If -bbb is specified, a blank line is forced before every block comment. Default: -nbbb.

-bc,-nbc If -bc is specified, then a newline is forced after each comma in a declaration. -nbc turns off this option. The default is -nbc.

-br,-bl Specifying -bl lines up compound statements like this:

```
if (...)
{
code
}
```

Specifying -br (the default) makes them look like this:

```
if (...) {
code
}
```

-c*n* The column in which comments on code start. The default is 33.

-cd*n* The column in which comments on declarations start. The default is for these comments to start in the same column as those on code.

-cdb,-ncdb Enables (disables) the placement of comment delimiters on blank lines. With this option enabled, comments look like this:

```
/*
 * this is a comment
 */
```

Rather than like this:

```
/* this is a comment */
```

This only affects block comments, not comments to the right of code. The default is -cdb.

-ce,-nce Enables (disables) forcing 'else's to cuddle up to the immediately preceding '}'. The default is -ce.

-ci*n* Sets the continuation indent to be n. Continuation lines will be indented that far from the beginning of the first line of the statement. Parenthesized expressions have extra indentation added to indicate the nesting, unless -lp is in effect. -ci defaults to the same value as -i.

-cli*n* Causes case labels to be indented n tab stops to the right of the containing switch statement. -cli0.5 causes case labels to be indented half a tab stop. The default is -cli0. (This is the only option that takes a fractional argument.)

-d*n* Controls the placement of comments which are not to the right of code. Specifying -d1 means that such comments are placed one indentation level to the left of code. The default -d0 lines up these comments with the code. See the section on comment indentation below.

-di*n* Specifies the indentation, in character positions, from a declaration keyword to the following identifier. The default is -di16.

-dj,-ndj -dj left justifies declarations. -ndj indents declarations the same as code. The default is -ndj.

-ei,-nei Enables (disables) special else-if processing. If enabled, ifs following elses will have the same indentation as the preceding if statement. The default is -ei.

-fc1,-nfc1 Enables (disables) the formatting of comments that start in column 1. Often, comments whose leading '/' is in column 1 have been carefully hand formatted by the programmer. In such cases, -nfc1 should be used. The default is -fc1.

-i*n* The number of spaces for one indentation level. The default is 8.

-ip,-nip Enables (disables) the indentation of parameter declarations from the left margin. The default is -ip.

-l*n* Maximum length of an output line. The default is 78.

-lp,-nlp Lines up code surrounded by parenthesis in continuation lines. If a line has a left parenthesis which is not closed on that line, then continuation lines will be lined up to start at the character position just after the left parenthesis. For example, here is how a piece of continued code looks with -nlp in effect:

```
p1 = firstprocedure(secondprocedure(p2, p3),
   thirdprocedure(p4, p5));
```

With -lp in effect (the default) the code looks somewhat clearer:

```
p1 = firstprocedure(secondprocedure(p2, p3),
                    thirdprocedure(p4, p5));
```

Inserting two more newlines we get:

```
p1 = firstprocedure(secondprocedure(p2,
                                    p3),
                    thirdprocedure(p4,
                                   p5));
```

-npro Causes the profile files, ./.indent.pro and ~/.indent.pro, to be ignored.

-pcs,-npcs If true (-pcs) all procedure calls will have a space inserted between the name and the '('. The default is -npcs.

-ps,-nps If true (-ps) the pointer following operator '->' will be surrounded by spaces on either side. The default is -nps.

-psl,-npsl If true (-psl) the names of procedures being defined are placed in column 1 their types, if any, will be left on the previous lines. The default is -psl.

-sc,-nsc Enables (disables) the placement of asterisks ('*'s) at the left edge of all comments. The default is -sc.

-sob,-nsob If -sob is specified, indent will swallow optional blank lines. You can use this to get rid of blank lines after declarations. Default: -nsob.

-st Causes indent to take its input from stdin, and put its output to stdout.

-T*typename* Adds typename to the list of type keywords. Names accumulate: -T can be specified more than once. You need to specify all the typenames that appear in your program that are defined by typedefs nothing will be harmed if you miss a few, but the program won't be formatted as nicely as it should. This sounds like a painful thing to have to do, but it's really a symptom of a problem in C: typedef causes a syntactic change in the language and indent can't find all typedefs.

-troff Causes indent to format the program for processing by troff. It will produce a fancy listing in much the same spirit as vgrind. If the output file is not specified, the default is standard output, rather than formatting in place.

-v,-nv -v turns on verbose mode; -nv turns it off. When in verbose mode, indent reports when it splits one line of input into two or more lines of output, and gives some size statistics at completion. The default is -nv.

USER-DEFINED PROFILE

You may set up your own profile of defaults to indent by creating a file called .indent.pro in either your login directory and/or the current directory and including whatever switches you like. Switches in .indent.pro in the current directory override those in your login directory (with the exception of -T type definitions, which just accumulate). If indent is run and a profile file exists, then it is read to set up the program's defaults. The switches should be separated by spaces, tabs or newlines. Switches on the command line, however, override profile switches.

COMMENTS

Box comments: Indent assumes that any comment with a dash or star immediately after the start of comment (that is, '/*-' or '/**') is a comment surrounded by a box of stars. Each line of such a comment is left unchanged, except that its indentation may be adjusted to account for the change in indentation of the first line of the comment.

Straight text: All other comments are treated as straight text. Indent fits as many words (separated by blanks, tabs, or newlines) on a line as possible. Blank lines break paragraphs.

Comment indentation

If a comment is on a line with code it is started in the 'comment column', which is set by the -cn command line parameter. Otherwise, the comment is started at *n* indentation levels less than where code is currently being placed, where n is specified by the -dn command line parameter. If the code on a line extends past the comment column, the comment starts further to the right, and the right margin may be automatically extended in extreme cases.

PREPROCESSOR LINES

In general, indent leaves preprocessor lines alone. The only reformatting that it will do is to straighten up trailing comments. It leaves embedded comments alone. Conditional compilation (#ifdef...#endif) is recognized and indent attempts to correctly compensate for the syntactic peculiarities introduced.

C SYNTAX

Indent understands a substantial amount about the syntax of C, but it has a forgiving parser. It attempts to cope with the usual sorts of incomplete and ill-formed syntax. In particular, the use of macros like:

```
#define forever for(;;)
```

is handled properly.

FILES

./.indent.pro	profile file
~/.indent.pro	profile file

indxbib

build inverted index for bibliography (unsupported)

The indxbib command makes an inverted index to the named *databases* (or files) for use by lookbib and refer. Inverted index files contain bibliographic references (or other kinds of information) separated by blank lines.

---

indxbib *database*

DESCRIPTION

Indxbib is a shell script that calls `/usr/lib/refer/mkey` and `/usr/lib/refer/inv`. The first program, *mkey*, truncates words to six characters, and maps upper case to lower case. It also discards words shorter than three characters, words among the 100 most common English words, and numbers (dates) < 1900 or > 2000. These parameters can be changed (see the *Document Preparation Guide*). The second program, *inv*, creates an entry file (`.ia`), a posting file (`.ib`), and a tag file (`.ic`), all in the working directory.

A bibliographic reference is a set of lines, constituting fields of bibliographic information. Each field starts on a line beginning with a per cent sign (%), followed by a key-letter, then a blank, and finally the contents of the field, which can continue until the next line starting with a per cent sign.

It is possible to search multiple databases, as long as they have a common index made by indxbib. In that case, only the first argument given to indxbib is specified to lookbib.

FILES

x.ia, x.ib, x.ic, where x is the first argument, or if these are not present, x.ig, x

SEE ALSO

refer, addbib, sortbib, roffbib, lookbib
Document Preparation Guide: Refer — A Bibliography System

info

documentation reader

Info is a program for reading directories of hierarchical documention. It is virtually identical to the INFO program written at the MIT Artificial Intelligence Laboratory. Info is completely self-documenting, and contains a tutorial for novice users. Info also can be invoked from inside of *EMACS*.

---

info [*node...*]

FILES

`/usr/lib/info/*`

SEE ALSO

emacs
GNU Emacs Manual

install

install binaries

The install command moves a binary file to a specified destination.

---

install [*option...*] *binary destination*

ARGUMENTS

-c specifies that the binary is to be copied, rather than moved.

-m *mode* specifies a mode (default is 755).

-o *owner* specifies an owner (default is `root`).

-g *group* specifies a group (default is `staff`).

-s specifies that the binary is to be stripped after being installed.

binary specifies the binary file to move.

destination if *destination* already exists, it is removed before *binary* is moved. If the destination is a directory, *binary* is moved into the *destination* directory with its original filename.

DESCRIPTION

Install refuses to move a file onto itself.

The install program attempts to set the owner and group for the binary either as specified by the user or using the default values. An error message will be displayed only if the user explicitly requested an owner or group for the binary.

SEE ALSO

chgrp, chmod, cp, mv, strip
Concentrix-2800 System Administration: chown

jobs

list active jobs (C shell)

The jobs command displays a list of currently active jobs.

---

jobs [-l]

ARGUMENTS

-l lists the process identification number of each job. It also lists the working directories of suspended or background jobs when different from the current working directory.

DESCRIPTION

Jobs displays the job number, the status, and the name of the command that created the job. The current job is marked with a plus sign; the previous job is marked with a hyphen.

SEE ALSO

bg, fg

join

relational database operator

The join command forms, on the standard output, a join of the two relations specified by the lines of two files.

---

join [*option...*] *file1 file2*

ARGUMENTS

-a*n* produces, in addition to the normal output, a line for each unpairable line in file *n*, where *n* is 1 or 2.

-e *s* replaces empty output fields by the string *s*.

-j*n m* joins on the field *m* of file *n*. If *n* is missing, use the field *m* in each file.

-o *list* each output line comprises the fields specified in *list*, each element of which has the form *n.m*, where *n* is a file number and *m* is a field number.

-t*c* uses character *c* as a separator (tab character). Every appearance of *c* in a line is significant.

file1 specifies an input file. A hyphen specifies the standard input.

file2 specifes an input file.

DESCRIPTION

File1 and *file2* must be sorted in increasing ASCII collating sequence on the fields on which they are to be joined, normally the first in each line.

There is one line in the output for each pair of lines in *file1* and *file2* that have identical join fields. The output line normally consists of the common field, then the rest of the line from *file1*, then the rest of the line from *file2*.

Fields are normally separated by blank, tab or newline. In this case, multiple separators count as one, and leading separators are discarded.

NOTES

- With default field separation, the collating sequence is that of *sort -b*; with -t, the sequence is that of a plain sort.
- The conventions of join, sort, comm, uniq, look and awk are wildly incongruous.

SEE ALSO

sort, comm, awk

kconfig

system configuration and memory usage

The kconfig command gives a continuous display of system memory usage (swap, page and global), process statistics, or mbuf usage. Unless the –i option is used, the display is updated every 5 seconds. The version number of the Concentrix release and information about shared memory, daylight saving time, and NFS installation kernel variables are also displayed.

/usr/alliant/kconfig [bmp] [–i*n*]

ARGUMENTS

–b displays mbuf statistics.

–m displays memory statistics. This is the default display.

–p displays process statistics.

–i*n* (optional) causes the display to pause *n* seconds between display updates. The default is 5 seconds.

DESCRIPTION

Typing one of the option characters at any point switches to the appropriate display when kconfig is in continuous display mode. The time interval can be changed from 1 to 10 seconds by typing the appropriate character from 1 to 0 (0=10).

To quit, type q, Q, or CTRL C at any point.

EXAMPLE

The following example shows the display for the kconfig –p option:

```
% kconfig -p
    Alliant Concentrix 2800 2.1.0 on 'marley'       Version# 20000003
    Built Tue Dec 18 12:09:24 EST 1990              Thu Jan 17 16:37:12 1991
  context switches ..............  1305   number of swapped runnable procs     0
  idle switches .................     0   super. mode ce based exceptions     99
  number of sleeps causing switch   214   user mode ce based exceptions ..  2057
  calls to trap .................     0   collapse complex events ........     0
  calls to syscall() ............ 11003   expand complex events ..........     0
  asynchronous system traps ......  257   dynamic cluster explodes .......     4
  forks executed ................   55   dynamic reconfigs to cluster ...     4
  obreak system calls ...........  730   device interrupts ..............  1909
  exec system calls .............   52   pseudo-dma interrupts ..........     0
  exec system calls (command file)  0   kfe control block requests .....    12
  known file searches ...........    0   kfe control block hits .........     0
  known file hits ...............    0

 Concentrix will switch to daylight saving time
  Only ROOT can use setclass
  Remote console port    = DISABLED
  NFS                    = ENABLED
```

FILES

/dev/kmem

SEE ALSO

mon, ps, xproc

keylogin

decrypt and store secret key

The keylogin command prompts for the user's login password, and uses it to decrypt the user's secret key stored in the publickey database. Once decrypted, the user's key is stored by the local key server process keyserv to be used by any secure network services, such as NFS. Normally, login does this work when the user logs onto the system, but running keylogin may be necessary if the user did not type a password to login.

---

keylogin

SEE ALSO

chkey, login
Concentrix-2800 System Administation: keyserv, newkey
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: publickey
Concentrix Network File System User's Guide, chapter on NFS administration

kill

terminate a process (Bourne shell)

The kill command sends the TERM (terminate, 15) signal to the specified processes, which must belong to the current user unless he is the superuser.

---

kill [*-signal*] *pid...*

kill -l

ARGUMENTS

-signal specifies a signal name or number to be sent instead of terminate (see the system call *sigvec*).

pid specifies a process identification. By convention, if process number 0 is specified, all members in the process group (i.e. processes resulting from the current login) are signaled (this does not work with the C shell)

-l lists signal names as given in /usr/include/signal.h, stripped of the common SIG prefix.

DESCRIPTION

The terminate signal kills processes that do not catch the signal; kill -9 is a sure kill, as the KILL (9) signal cannot be caught.

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

NOTES

- For Concentrix-2800-2800 V.2.x, a thread_id may be used as a PID. Signals are delivered to the root thread by default, since the thread_id of the root thread is the same as its PID.

  If you send the signal KILL (for example, the command kill -9) to any thread of a process, the entire process is terminated.

SEE ALSO

ps
Concentrix-2800 System Reference: System Calls and C Library Subroutines: kill, sigvec

kill

terminate job or process (C shell)

The kill command sends the TERM (terminate, 15) signal to the specified processes or jobs, which must belong to the current user unless he is the superuser. Kill is built into the C shell.

---

kill [*-signal*] *pid...*

kill [*-signal*] *%job...*

kill -l

ARGUMENTS

-signal specifies a signal name or number to be sent instead of terminate (see the system call sigvec).

pid specifies a process identification. Negative process numbers may be used to send signals to more than one process. See kill(2) for details.

%job specifies a job number.

-l lists signal names as given in /usr/include/signal.h, stripped of the common SIG prefix.

DESCRIPTION

The terminate signal kills processes that do not catch the signal; `kill -9` is a sure kill, as the KILL (9) signal cannot be caught.

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

NOTES

- If the signal being sent is TERM (terminate) or HUP (hangup), the job or process is sent a CONT (continue) signal as well.
- For Concentrix-2800 V.2.x, a thread_id may be used as a PID. Signals are delivered to the root thread by default, since the thread_id of the root thread is the same as its PID.

  If you send the signal SIGKILL (for example, the command kill -9) to any thread of a process, the entire process is terminated.

SEE ALSO

jobs, ps
Concentrix-2800 System Reference: System Calls and C Library Subroutines: kill, sigvec
Concentrix-2800 System User's Guide: C Shell Reference

last

indicate last logins of users and teletypes

The last command displays a record of logins and logouts, in reverse order, showing users and teletypes, the time at which the session began, the duration of the session, and the teletype that the session took place on. If a session is still continuing or was cut short by a reboot or crash, *last* so indicates.

---

last [-n] [*name...*] [*tty...*]

ARGUMENTS

-n specifies how many lines to display (default is all available data).

-f specifies an alternate file to be examined (rather than /usr/adm/wtmp, the default).

name specifies a user name (default is all users).

tty specifies a teletype (default is all teletypes.). This can be given fully or abbreviated, for example: `0` is the same as `tty0`.

DESCRIPTION

If last is interrupted, it indicates how far the search has progressed in *wtmp*. If interrupted with a quit signal (generated by a CTRL\) last indicates how far the search has progressed so far, and the search continues.

EXAMPLE

```
last root console
```

lists all `root` sessions as well as all sessions on the console terminal. The pseudo-user reboot logs in at reboots of the system, thus

```
last reboot
```

gives an indication of mean time between reboots.

FILES

/usr/adm/wtmp	login data base
/usr/adm/shutdownlog	records shutdowns and reasons for same

SEE ALSO

lastcomm
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: wtmp
Concentrix-2800 System Administration: ac

lastcomm

show commands in reverse order

The lastcomm command displays information on previously executed commands. For each process entry, the following is displayed.

- The name of the user who ran the process.
- Flags, as accumulated by the accounting facilities in the system.
- The command name under which the process was called.
- The amount of cpu time used by the process (in seconds).
- The time the process exited.

---

lastcomm [*command...*] [*user...*] [*terminal...*]

ARGUMENTS

command specifies a command (default is all commands).

user specifies a user name (default is all users).

term specifies a terminal name (default is all terminals).

DESCRIPTION

The flags are encoded as follows:

S indicates the command was executed by the superuser.

F indicates the command ran after a fork, but without a following exec.

D indicates the command terminated with the generation of a *core* file.

X indicates the command was terminated with the signal SIGTERM.

EXAMPLE

```
lastcomm a.out root ttyd0
```

produces a listing of all the executions of commands named `a.out` by user `root` on the terminal `ttyd0`.

SEE ALSO

last, sigvec
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: core

ld

link editor

The ld command combines several object programs into one, resolves external references, and searches libraries. The output is left in a.out, which is made executable if no errors occurred during the load.

---

ld [*option...*] *file...*

ARGUMENTS

-A *name* specifies incremental loading, that is, linking is to be done in a manner so that the resulting object can be read into an already executing program. The *name* argument is the name of a file whose symbol table is taken as a basis on which to define additional symbols. Only newly linked material is entered into the text and data portions of a.out, but the new symbol table reflects every symbol defined before and after the incremental load. The *name* argument must appear before any other object file in the argument list. The -T option can be used as well, and is taken to mean that the newly linked segment commences at the corresponding address (which must be a multiple of 4096). The default value is the old value of _end.

-b *module* aligns the specified *module*'s data segment on a page boundary, converts all common block definitions to data segment references, and extends the segment size to the next page boundary.

-d forces definition of common storage even if the -r option is present.

-D *n* aligns the data segment to the first *n* byte boundary following the text segment. *n* is a hexadecimal number.

-e *entry* specifies the name of the entry point of the loaded program; the default is the start of the text section.

-i *file* searches the specified file to include any global symbols that are undefined after all previous object files have been searched. Thus, a program can access symbol definitions unique to another executable image without generating error messages. Undefined and multiply defined symbols within *file* are ignored.

-F *name n* Forces the origin of the section named *name* to be the hexadecimal address *n*.

-lx is an abbreviation for libx.a, where *x* is a string. Ld searches for the following files until it finds one that exists:

```
/lib/libx.a
/usr/lib/libx.a
/usr/local/lib/libx.a.
```

A library is searched when its name is encountered, so the placement of the -l option is significant. To override the search path, define an environment variable LD_PATH (see below) to contain a series of pathnames.

-m *symbol* defines a global *symbol* whose value coincides with the start of the data segment for the next module in the load sequence.

-M produces a primitive load map, listing the names of the files that are loaded.

-nc (no concurrency) forces an image that would normally run on the complex to run on a detached CE.

-o *name* specifies the name of the ld output file (default is a.out).

-r generates relocation bits in the output file so that it can be the subject of another ld run. This option also prevents final definitions from being given to common symbols, and suppresses the "undefined symbol" diagnostics.

-s strips the output, that is, removes the symbol table and relocation bits to save space (but impairs the usefulness of the debuggers). The same function is served by strip.

-S strips the output by removing all symbols except locals and globals.

-t (trace) displays the name of each file as it is processed.

-T *n* sets the text segment origin to the hexadecimal number *n*. The default origin is 0x80000.

-u *symbol* enters *symbol* as undefined in the symbol table. This option is useful for loading wholly from a library, since initially the symbol table is empty and an unresolved reference is needed to force the loading of the first routine.

-X saves local symbols except for those whose names contain a ".". This option is used by cc to discard internally-generated labels while retaining symbols local to routines.

-x prohibits preservation of local (non-.globl) symbols in the output symbol table; only allows external symbols to be entered. This option saves some space in the output file.

-y*sym* indicates each file in which *sym* appears, its type and whether the file defines or references it. Many such options can be given to trace many symbols. (It is usually necessary to begin *sym* with an underscore, as external C, FORTRAN and Pascal variables begin with underscores.)

file specifies an object program. If several object *files* are given, ld combines them, producing an object module that can be executed or can become the input for a further ld run. (In the latter case, the -r option must be given to preserve the relocation bits.)

DESCRIPTION

The argument routines are concatenated in the order specified. The entry point of the output is the beginning of the first routine (unless the -e option is specified).

If any argument is a library, it is searched exactly once at the point it is encountered in the argument list. Only those routines defining an unresolved external reference are loaded.

If you have an environment variable named LD_PATH, whose value is a search path, (a list of pathnames separated by spaces or tabs), then whenever ld needs a library, it looks for it first using the first path in the search list. If not found, it searches the second path specified, and so forth. If LD_PATH is not defined, ld looks for libraries first in /lib, then in /usr/lib, and finally in /usr/local/lib.

If you have a set of private libraries that you wish searched whenever you load some object files, you can put them in a private directory and then set LD_PATH to point to the standard list followed (or preceded) by your private area. For example:

```
% setenv LD_PATH "/lib /usr/lib /usr/local/lib /usr/smith/lib"
```

If a routine from a library references another routine in the library, and the library has not been processed by ranlib, the referenced routine must appear after the referencing routine in the library. Thus the order of programs within libraries may be important. The first member of a library should be a file named __.SYMDEF, that is understood to be a dictionary for the library as produced by ranlib; the dictionary is searched iteratively to satisfy as many references as possible.

The symbols _etext, _edata and _end (etext, edata and end in C) are reserved, and if referred to, are set to the first location above the program, the first location above initialized data, and the first location above all data respectively. It is erroneous to define these symbols.

The Concentrix-2800 loader diagnoses the use of different sized global data areas (COMMON blocks in Fortran terminology) during the link process. When the linker encounters global data areas of different sizes in different modules, it uses the largest size, *as long as no initialized data exists in the global data area.*

If initialized data exists in a declaration of a global data area, the size of that declaration is used. If another declaration of the global data area specifies a larger size, the loader prints the following error:

Symbol "*symbol-name*" has been defined as size *n* but is referred to as size *m*

FILES

/lib/lib*.a	libraries
/usr/lib/lib*.a	more libraries
/usr/local/lib/lib*.a	still more libraries
a.out	output file

SEE ALSO

as, ar, cc, ranlib

leave

remind you when you have to leave

The leave command waits until the specified time, then reminds you that you have to leave. You are reminded five minutes and one minute before the actual time, at the time, and every minute thereafter. When you log out, leave exits just before it would have displayed the next message.

---

leave [[+]*hhmm*]

ARGUMENTS

+ specifies an offset in hours and minutes from the current time.

hh specifies a time in hours (on a 12 or 24 hour clock). Times are converted to a 12 hour clock, and assumed to be in the next 12 hours.

mm specifies a time in minutes.

If you do not supply an argument, leave prompts with `When do you have to leave?`. A newline causes leave to exit, otherwise the reply is assumed to be a time. This form is suitable for inclusion in a .login or .profile.

DESCRIPTION

Leave ignores interrupts, quits, and terminates. To get rid of it, either log out or use kill -9 giving its process id.

SEE ALSO

calendar

lex

generator of lexical analysis programs

The lex command generates programs to be used in simple lexical analyis of text. The input files contain regular expressions to be searched for, and actions written in C to be executed when expressions are found.

---

lex [-tvfh] [*file...*]

ARGUMENTS

-t places the result on the standard output instead of in file `lex.yy.c`.

-v displays a one-line summary of statistics of the generated analyzer.

-f compiles faster: does not pack the resulting tables; limited to small programs.

-n suppresses statistics (default).

file specifies an input file (standard input default).

DESCRIPTION

A C source program, `lex.yy.c` is generated, to be compiled thus:

```
cc lex.yy.c -ll
```

This program, when run, copies unrecognized portions of the input to the output, and executes the associated C action for each regular expression that is recognized.

EXAMPLE

```
lex lexcommands
```

would draw lex instructions from the file lexcommands, and place the output in lex.yy.c.

```
%%
[A-Z] putchar(yytext[0]+'a'-'A');
[ ]+$
[ ]+ putchar(' ');
```

is an example of a lex program that would be put into a lex command file. This program converts upper case to lower, removes blanks at the end of lines, and replaces multiple blanks by single blanks.

SEE ALSO

yacc, sed
Concentrix Programming Tools Guide: LEX - Lexical Analyzer Generator

limit

limit resource consumption (C shell)

The limit command prevents the current process and each process it creates from individually exceeding a specified limit on one or more system resources.

---

limit [-h] [*resource* [*maximum*]]

ARGUMENTS

-h use the hard limits instead of the current limits. The hard limits impose a ceiling on the values of the current limits. Only the superuser may raise the hard limits, but a user may lower or raise the current limits within the legal range.

resource is one of:

cputime the maximum number of cpu-seconds to be used by each process.

filesize the largest single file that can be created.

datasize the maximum growth of the data+stack region via the sbrk system call beyond the end of the program text.

stacksize the maximum size of the automatically-extended stack region.

coredumpsize
the size of the largest core dump that will be created.

memoryuse
The maximum size, in bytes, a process' resident set size can grow to. This imposes a limit on the amount of physical memory to be given to a process; if memory is tight, the system prefers to take memory from processes that are exceeding their declared resident set size.

if *resource* is omitted, all limitations are given.

maximum is a number (floating point or integer) followed by a scale factor. For all resources other than cputime the default scale is k or kilobytes (1024 bytes); a scale factor of m or megabytes can also be used. For cputime the default scaling is seconds, while m for minutes or h for hours, or a time of the form mm:ss giving minutes and seconds can be used. If *maximum* is omitted, the current limit is printed.

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

EXAMPLE

```
% limit
cputime         unlimited
filesize        unlimited
datasize        65536 kbytes
stacksize       512 kbytes
coredumpsize    20480 kbytes
memoryuse       unlimited
```

SEE ALSO

unlimit
Concentrix-2800 System Reference: System Calls and C Routines: getrlimit, setrlimit

lint

C program verifier

The lint command attempts to detect features of C program files that are likely to be bugs, non-portable, or wasteful. It also checks the type usage of the program more strictly than the compiler.

---

lint [-abchnpuvx] ***file...***

ARGUMENTS

-a reports assignments of long values to int variables.

-b reports *break* statements that cannot be reached. (This is not the default because, unfortunately, most **lex** and many **yacc** outputs produce dozens of such comments.)

-c complains about casts that have questionable portability.

-h applies a number of heuristic tests to attempt to intuit bugs, improve style, and reduce waste.

-n suppresses compatibility checking against the standard library.

-p attempts to check portability to the *IBM* and *GCOS* dialects of C.

-u suppresses complaints about functions and variables used and not defined, or defined and not used (suitable for running *lint* on a subset of files out of a larger program).

-v suppresses complaints about unused arguments in functions.

-x reports variables referred to by extern declarations, but never used.

-z suppresses complaints about structures that are never defined (e.g. using a structure pointer without knowing its contents.).

file specifies an input file. By default, it is assumed that all the *files* are to be loaded together; they are checked for mutual compatibility.

Function definitions for certain libraries are available to lint; the libraries are referred to by a conventional name, such as -lm, in the style of ld. Arguments ending in .ln are also treated as library files. To create lint libraries, use the -C option:

```
lint -Cfoo files . . .
```

where *files* are the C sources of library foo. The result is a file llib-lfoo.ln in the correct library format suitable for linting programs using foo.

The -D, -U, and -I options of cc are also recognized as separate arguments.

DESCRIPTION

Among the things that are currently found are unreachable statements, loops not entered at the top, automatic variables declared and not used, and logical expressions whose value is constant. Moreover, the usage of functions is checked to find functions that return values in some places and not in others, functions called with varying numbers of arguments, and functions whose values are not used.

The system call exit and other functions that do not return are not understood and thus cause erroneous messages.

Certain conventional comments in the C source change the behavior of lint:

- `/*NOTREACHED*/` at appropriate points stops comments about unreachable code.
- `/*VARARGSn*/` suppresses the usual checking for variable numbers of arguments in the following function declaration. The data types of the first *n* arguments are checked; a missing *n* is taken to be 0.
- `/*ARGSUSED*/` turns on the –v option for the next function.
- `/*LINTLIBRARY*/` at the beginning of a file shuts off complaints about unused functions in this file.

NOTES

- There are some things for which you cannot suppress diagnostics.
- The /*NOSTRICT*/ comment (which is supposed to turn off strict type checking in the next statement) is not implemented in this version.

FILES

/usr/lib/lint/lint[12]	programs
/usr/lib/lint/llib–lc.ln	declarations for standard functions
/usr/lib/lint/llib–lc	human readable version of above
/usr/lib/lint/llib–port.ln	declarations for portable functions
/usr/lib/lint/llib–port	human readable . . .
llib–l*.ln	library created with –C

SEE ALSO

Concentrix Programming Tools Guide: Lint, a C Program Checker

ln

make links

The ln command creates links. A link is a directory entry referring to a file; the same file (together with its size, all its protection information, etc.) can have several links to it.

---

ln [**-s**] *file*

ln [**-s**] *file link*

ln [**-s**] *file... directory*

ARGUMENTS

-s	creates symbolic links (ln creates hard links by default).
file	specifies the pathname of an existing file.
link	specifies the pathname of the desired link to *file*.
directory	specifies the directory in which to create links.

DESCRIPTION

There are two kinds of links: hard links and symbolic links. A hard link to a file is indistinguishable from the original directory entry; any changes to a file are effective independent of the name used to reference the file. Hard links cannot span file systems and cannot refer to directories.

A symbolic link contains the name of the file to which it is linked. The referenced file is used when an open operation is performed on the link. A stat on a symbolic link returns the linked-to file; an lstat must be done to obtain information about the link. The readlink call can be used to read the contents of a symbolic link. Symbolic links can span file systems and can refer to directories.

With one argument, the argument should be the name of a file in another directory. In this case, ln creates a link in the current directory to the file you named. The name of the link is the last part of the filename.

Given two arguments (without the second one being a directory), ln creates a link to the first argument and names the link with the second argument.

Given two or more arguments, where the last argument is a directory, ln creates a series of links in that directory. All the filenames mentioned are given a link in that directory, with the link name being the last part of the filename.

SEE ALSO

rm, cp, mv, link, readlink, stat, symlink

lock

reserve terminal

The lock command requests a password from the user, then locks the terminal from other users.

---

lock [*-number*]

ARGUMENTS

-number the timeout period for lock, in minutes

DESCRIPTION

The lock command requests a password from the user, then locks the terminal from other users.

There are four conditions under which the terminal will be unlocked.

- the user types the password again
- the timeout period expires
- the password for root is entered
- the process is killed by someone with appropriate permission

logger

make entries in the system log

Logger provides a command line interface to the syslog(3) system log module.

---

logger [**-t** *tag*] [**-p** *pri*] [**-i**] [**-f** *file*] [*message ...*]

ARGUMENTS

-t *tag* Mark every line in the log with the specified *tag*.

-p *pri* Enter the message with the specified priority. The priority may be specified numerically or as a "facility.level" pair. For example, "-p local3.info" logs the message(s) as informational level in the local3 facility. The default is "user.notice."

-i Log the process id of the logger process with each line.

-f *file* Log the specified file.

message The message to log; if not specified, the **-f** *file* or standard input is logged.

DESCRIPTION

A message can be given on the command line, which is logged immediately, or a file is read and each line is logged.

EXAMPLES

```
logger System rebooted

logger -p local0.notice -t HOSTIDM -f /dev/idmc
```

SEE ALSO

syslog
Concentrix-2800 System Administration: syslogd

login

create new user process

The login command replaces your login shell with /bin/login. It allows you to change users on a dialup line without having to re-establish the connection. Login is recognized by sh and csh and executed directly (without forking).

---

login [**-p**] [*name*]

login [**-f**] [*name*]

login **-h** *hostname*

login [*name*]

ARGUMENTS

-p preserves the entire user environment.

name specifies the name of a system user. If omitted, login asks for a user name.

-h *hostname*
specifies telnetd, etc. For superuser use only.

-f *name* Specifies preauthenticated login: datakit, xterm, etc.

DESCRIPTION

If appropriate, login asks for a password. Echoing is turned off (if possible) during the typing of the password, so that it does not appear on the written record of the session.

After a successful login, accounting files are updated and you are informed of the existence of mail. The message of the day is displayed, as is the time you last logged in unless you have a `.hushlogin` file in your home directory (mostly used by programs such as uucp).

Login initializes the user and group IDs and the working directory, then executes a command interpreter according to specifications found in a password file. Argument 0 of the command interpreter is the name of the command interpreter with a leading hyphen prepended (usually "-csh").

Login also initializes the user environment with information specifying home directory, command interpreter, terminal type (if available) and user name. If -p is specified, the remainder of the environment is preserved, otherwise it is discarded.

Before executing the command interpreter, login establishes the user's default scheduling classes. The values are taken from the file /etc/usrclass. In the absence of /usr/usrclass or an entry in this file for the user, the system defaults apply. The system defaults are 1 (concurrent CE), 2 (detached CE), 3 (any resource), and 4 (IP only).

If the file /etc/nologin exists, login displays its contents on your terminal and exits. This prevents users from logging in when the system is about to go down.

DIAGNOSTICS

- `Login incorrect` – the name or the password is bad.
- `No Shell` – consult your system administrator.
- `cannot open password file` – consult your system administrator.
- `no directory` – consult your system administrator.

NOTES

- An undocumented option, –r is used by the remote login server, rlogind *(Concentrix-2800 or Concentrix System Administration)* to force login to enter into an initial connection protocol. Telnetd and other servers use the –h option to list the host from which the connection was received.

FILES

/etc/utmp	accounting
/usr/adm/wtmp	accounting
/usr/spool/mail/*	mail
/etc/motd	message-of-the-day
/etc/passwd	password file
/etc/nologin	stops logins
.hushlogin	makes login quieter

SEE ALSO

mail, passwd, rlogin
Concentrix-2800 System Administration: init, getty, shutdown
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: passwd

login

execute login arguments (Bourne shell)

The login special command is equivalent to executing a login.

---

login [*arg...*]

NOTES

- Equivalent to 'execute login argument...'

SEE ALSO

login

logout

terminate login shell (C shell)

The logout command terminates a login shell and drops dialup connections.

---

logout

NOTES

- Use logout to log out if the C shell variable *ignoreeof* is set.

SEE ALSO

login

look

find lines in a sorted list

The look command displays all lines in a sorted file that begin with a specific string. It uses binary search.

---

look [-df] *string* [*file*]

DESCRIPTION

The options d and f affect comparisons as in sort:

-d specifies "dictionary" order: only letters, digits, tabs and blanks participate in comparisons.

-f specifies "fold." Upper case letters compare equal to lower case.

string is the target string.

file specifies an input file. If omitted, /usr/dict/words is assumed with collating sequence -df.

FILES

/usr/dict/words

SEE ALSO

sort, grep

lookbib

find references in a bibliography (unsupported)

The lookbib command uses an inverted index made by indxbib to find sets of bibliographic references. It reads keywords typed after the right angle bracket (>) prompt on the terminal, and retrieves records containing all these keywords. If nothing matches, nothing is returned except another prompt.

---

lookbib [-n] *database*

ARGUMENTS

-n do not ask the user if he or she needs instructions.

database specifies the name of the database to look in.

DESCRIPTION

A bibliographic reference is a set of lines, constituting of fields of bibliographic information. Each field starts on a line beginning with a per cent sign (%), followed by a key-letter, then a blank, and finally the contents of the field, which can continue until the next line starting with a per cent sign.

It is possible to search multiple databases, as long as they have a common index made by indxbib. In that case, only the first argument given to indxbib is specified to lookbib.

If lookbib does not find the index files (the .i[abc] files), it looks for a reference file with the same name as the argument, without the suffixes. It creates a file with a .ig suffix, suitable for use with fgrep. It then uses the fgrep file to find references. This method is simpler to use, but the .ig file is slower to use than the .i[abc] files, and does not allow the use of multiple reference files.

NOTES

- Probably all dates should be indexed, since many disciplines refer to literature written in the 1800s or earlier.

FILES

x.ia	
x.ib	
x.ic	where *x* is the first argument
x.ig	if those above are not present
x	

SEE ALSO

refer, addbib, sortbib, roffbib, indxbib
Document Preparation Guide: Refer – A Bibliography System

lorder

find ordering relation for object library

The lorder command displays a list of pairs of object file names. The first file of the pair refers to external identifiers defined in the second. The output can be processed by *tsort* to find an ordering of a library suitable for one-pass access by ld.

---

lorder *file...*

ARGUMENTS

file specifies an object or library archive (see ar) file.

EXAMPLE

This command builds a new library from existing .o files.

```
% ar cr library.a `lorder *.o | tsort`
% ranlib library.a
```

NOTES

- The names of object files, in and out of libraries, must end with .o.

FILES

*symref
*symdef
nm
sed
sort
join

SEE ALSO

tsort, ld, ar, ranlib

lpq

spool queue examination program

The lpq command reports the status of line printer jobs associated with a user. It examines the spooling area used by lpd *(Concentrix-2800 or Concentrix System Administration)*.

---

lpq [+[*n*]] [-l] [-P*pr*] [*job...*] [*user...*]

ARGUMENTS

+*n* displays the spool queue until it empties. A number immediately after the plus sign indicates that lpq should sleep *n* seconds in between scans of the queue.

-l displays information about each of the files comprising the job. Normally, only as much information as can fit on one line is displayed.

-P*pr* specifies a particular printer, otherwise the default line printer is used (or the value of the PRINTER variable in the environment).

All other arguments supplied are user names or job numbers to filter out only those jobs of interest.

DESCRIPTION

For each job submitted (i.e. invocation of lpr) lpq reports the user's name, current rank in the queue, the names of files comprising the job, the job identifier (a number that can be supplied to lprm for removing a specific job), and the total size in bytes.

Job ordering is dependent on the algorithm used to scan the spooling directory and is supposed to be FIFO (First-in First-Out). File names comprising a job may be unavailable (when lpr is used as a sink in a pipeline) in which case the file is indicated as `(standard input)`.

If lpq warns that there is no daemon present (i.e. due to some malfunction), the lpc *(Concentrix-2800 System Administration)* command can be used to restart the printer daemon.

NOTES

- Due to the dynamic nature of the information in the spooling directory *lpq* may report unreliably: Output formatting is sensitive to the line length of the terminal; this can result in widely spaced columns.

DIAGNOSTICS

- Unable to open various files.
- The lock file being malformed.
- Garbage files when there is no daemon active, but files in the spooling directory.

FILES

/etc/termcap	for manipulating the screen for repeated display
/etc/printcap	to determine printer characteristics
/usr/spool/*	the spooling directory, as determined from printcap
/usr/spool/*/cf*	control files specifying jobs
/usr/spool/*/lock	the lock file to obtain the currently active job

SEE ALSO

lpr, lprm
Concentrix-2800 System Administration: lpc, lpd

lpr

print files on line printer

The lpr command uses a spooling daemon to print the named files when facilities become available.

---

lpr [*option...*] [*name...*]

ARGUMENTS

-P*printer* forces output to a specific printer. Normally, the default printer (site dependent) or the value of the environment variable PRINTER is used.

-#*num* specifies the number of copies desired of each file named.

-C*class* specifies a job classification for use on the burst page.

-J*job* specifies the job name to print on the burst page. Normally, the first file's name is used.

-T*title* specifies the title used by pr instead of the file name.

-i*numcols* causes the output to be indented. If the next argument is numeric, it is used as the number of blanks to be printed before each line; otherwise, eight characters are printed.

-1234*font* specifies a font to be mounted on font position *i*. The daemon constructs a `.railmag` file referencing `/usr/lib/vfont/name.size`.

-w*num* specifies the page width for pr.

The following single letter options notify the line printer spooler that the files are not standard text files. The spooling daemon uses the appropriate filters to print the data accordingly.

-p uses *pr* to format the files (equivalent to print).

-l uses a filter that allows control characters to be printed and suppresses page breaks.

-t the files are assumed to contain data from troff (cat phototypesetter commands).

-n the files are assumed to contain data from ditroff (device independent troff).

-d the files are assumed to contain data from *tex* (DVI format from Stanford).

-g the files are assumed to contain standard plot data as produced by the *plot* subroutines (see *plot* for the filters used by the printer spooler).

-v the files are assumed to contain a raster image for devices like the Benson Varian.

-c the files are assumed to contain data produced by cifplot.

-f uses a filter that interprets the first character of each line as a standard Fortran carriage control character.

The remaining single-letter options have the following meanings.

-r removes the file upon completion of spooling or upon completion of printing (with the -s option).

-m sends mail upon completion.

-h suppresess the printing of the burst page.

-s uses symbolic links. Usually files are copied to the spool directory. The -s option uses symlink *(System Calls)* to link data files rather than trying to copy them so large files can be printed. This means the files should not be modified or removed until they have been printed.

name specifies a file to print. If omitted, the standard input is assumed.

EXAMPLE

```
lpr -C EECS foo.c
```

causes the system name (the name returned by *hostname*) to be replaced on the burst page by EECS, and the file foo.c to be printed.

```
lpr -#3 foo.c bar.c more.c
```

results in three copies of the file foo.c, followed by three copies of the file bar.c, etc.

```
cat foo.c bar.c more.c | lpr -#3
```

prints three copies of the concatenation of the files.

DIAGNOSTICS

- If you try to spool too large a file, it is truncated.
- Lpr objects to printing binary files.
- If a user other than root prints a file and spooling is disabled, lpr displays a message saying so and does not put jobs in the queue.
- If a connection to lpd on the local machine cannot be made, lpr says that the daemon cannot be started.
- Diagnostics may be printed in the daemon's log file regarding missing spool files by lpd.

NOTES

- Fonts for troff and tex reside on the host with the printer. It is currently not possible to use local font libraries.
- The Alliant version of this command prints two banner pages.

FILES

/etc/passwd	personal identification
/etc/printcap	printer capabilities data base
/usr/lib/lpd*	line printer daemons
/usr/spool/*	directories used for spooling
/usr/spool/*/cf*	daemon control files
/usr/spool/*/df*	data files specified in `cf` files
/usr/spool/*/tf*	temporary copies of `cf` files

SEE ALSO

lpq, lprm, pr ,symlink
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: printcap
Concentrix-2800 System Administration: lpc, lpd

lprm

remove jobs from line printer queue

The lprm command removes a job, or jobs, from a printer's spool queue. Lprm announces the names of any files it removes and is silent if there are no jobs in the queue that match the request list.

---

lprm [**-P***printer*] [-] [*job...*] [*user...*]

ARGUMENTS

-P*printer* specifies the queue associated with a specific printer (otherwise the default printer, or the value of the PRINTER variable in the environment is used).

- causes lprm to remove all jobs that a user owns. If the superuser employs this option, the spool queue is emptied entirely. The owner is determined by the user's login name and host name on the machine where the lpr command was invoked.

job specifies a job number (currently active job by default).

user specifying a user's name, or list of user names, causes lprm to attempt to remove any jobs queued belonging to that user (or users). This form of invoking lprm is useful only to the superuser.

DESCRIPTION

A user can dequeue an individual job by specifying its job number. This number can be obtained from the lpq program, as shown below:

```
% lpq -l
1st: ken                        [job #013ucbarpa]
(standard input)                100 bytes
% lprm 13
```

Lprm kills off an active daemon, if necessary, before removing any spooling files. If a daemon is killed, a new one is automatically restarted upon completion of file removals.

DIAGNOSTICS

- `Permission denied` – if the user tries to remove files other than his own.

NOTES

- The spooling directory is protected from users; thus lprm is normally the only method by which a user can remove a job.
- Since there are race conditions possible in the update of the lock file, the currently active job may be incorrectly identified.

FILES

/etc/printcap	printer characteristics file
/usr/spool/*	spooling directories
/usr/spool/*/lock	lock file used to obtain the pid of the current daemon and the job number of the currently active job

SEE ALSO

lpr, lpq
Concentrix-2800 System Administration: lpd

lprof

line level profiler

The lprof command is a profiling tool that gives an indication of which source lines in a program are consuming cpu time. It uses the gmon.out file generated for use by gprof, and symbol tables generated by the compilers (via the -g option) for use by dbx. The default display shows the source file, routine name, line number, number of seconds, and percentage of cpu time for each line with a non-zero tick count.

---

lprof [*image* [*profile*] [*option...*]

ARGUMENTS

image specifies the image file used to generate the profile. The symbol table information in the file must be augmented by the line number information generated by the -g compiler option. If no *image* is specified lprof attempts to open the file ./a.out.

profile specifies the file produced by profiling. If no *profile* is specified, lprof attempts to open the file ./gmon.out.

-f *routine* forces lprof to profile only the specified routine. You can specify several routines, each of which requires a separate -f option. The -f option overides the -e option.

-e *routine* forces lprof to exclude the specified routine from the profile. You can specify several routines, each of which requires a separate -e option.

-l *thresh* prints only lines with percentage of cpu time above *thresh*. The default is to print all lines with non-zero tick counts.

-a *thresh* prints in the profile the assembly code corresponding to each line with percentage of cpu time above *thresh*. To the left of each instruction is printed the number of profiling ticks in the bin associated with that instruction. By default no assembly code is printed.

-g disables the replacement of address values with global symbols during the display of assembly code, which is the default.

-s attempts to replace address values with the names of source routines and line numbers during the display of assembly code.

-h *hertz* specifies the specified *hertz* value to convert raw tick counts from the *profile* to seconds. The default value is obtained from the running system and is typically 100.0 hertz.

DESCRIPTION

The line level profiler, lprof, allows you to generate a profile of a routine on a line-by-line basis, taking information from existing gmon.out files. The profiling information is

actually taken to four-byte resolution, thus each line of source has a number of histogram "bins". This is a very useful tool if a time consuming subroutine has many loops and the branching makes it difficult to tell which loops are executed most often.

Programs to be profiled by lprof should be compiled with the -g and -pg compiler and linker switches. When the program exits normally, the file gmon.out is produced in the current working directory.

EXAMPLE

The command:

```
lprof a.out gmon.out
```

produces the output shown below:

```
foo.f                foo                   84          0.01 sec     0.3%
foo.f                foo                  120          0.01 sec     0.3%
foo.f                foo                  121          0.06 sec     2.0%
foo.f                foo                  131          0.01 sec     0.3%
foo.f                foo                  132          0.07 sec     2.3%
foo.f                foo                  133          0.01 sec     0.3%
foo.f                foo                  135          0.01 sec     0.3%
```

The first 20 characters show the name of the source file, as it was known to the compiler. The next 20 characters show the name of the routine. The third field shows the line number within the source file. The fourth field shows the number of seconds this line took to execute. The last field shows the percentage of all routines considered in the line level profile.

NOTES

- To obtain a profile sorted by cpu time:

  ```
  lprof a.out gmon.out | tee ll.prof | sort -r -n +3 > ll.sorted
  ```

- Global optimization efforts of the compiler may split a line into two or more sections, or may make it disappear altogether.

- Since the profiling bins are not aligned with the assembly instructions or source lines, a bin may be charged to the wrong line or the wrong instruction.

FILES

./a.out	opened by default to get line number information.
./gmon.out	opened by default to get profiling information.

SEE ALSO

dbx, gprof
Concentrix Programming Tools Guide: gprof
FX/2800-C or *FX/2800-Fortran Programmer's Handbook:* appendix on dbx

lptest

generate line printer ripple pattern

The lptest command writes the traditional "ripple test' pattern on standard output.

---

lptest [*length* [*count*]]

ARGUMENTS

length specifies the output line length. Default is 79.

count specifies the number of output lines to be generated. Default is 200.

DESCRIPTION

Lptest writes the traditional ripple test pattern on standard output. In 96 lines, this pattern will print all 96 printable ASCII characters in each position.

While originally created to test printers, it is quite useful for testing terminals, driving terminal ports for debugging purposes, or any other task where a quick supply of random data is needed.

If *count* is specified, l*ength* must be specified also.

ls

list contents of directory

The ls command displays the contents of the current directory. If you specify a directory, *ls* displays its contents. If you specify a file, ls displays its name and any other information requested. By default, ls output is sorted alphabetically.

---

ls [-acdefgilqrstulACLFR] ***name...***

ARGUMENTS

-a lists all entries, including those whose names begin with a period.

-c uses time of last edit (or last mode change) for sorting or printing.

-d displays the name of a directory argument, rather than its contents (often used with -l to get the status of a specific directory).

-e sorts the entries by their filename extension (the extension is taken to be the characters after the last period in the filename).

-f forces each argument to be interpreted as a directory, lists the name found in each slot, turns off -l, -t, -s, and -r, turns on -a, and suppresses sorting.

-g includes the group ownership of the file in a long output.

-i displays the i-number of each file in the first column of the report.

-l lists in long format, giving mode, number of links, owner, size in bytes, and time of last modification for each file (see below). If the file is a special file the size field instead contains the major and minor device numbers. If the file is a symbolic link to a directory, the pathname of the linked-to file is preceded by ->. If the file is a symbolic link to a file (softlink), @ precedes the linked-to file.

-q forces display of non-graphic characters in file names as a question mark (?); this is the default when output is to a terminal.

-r reverses the order of the sort to display reverse alphabetic or oldest first as appropriate.

-s displays the size in kilobytes of each file.

-t sorts by time modified (latest first) instead of by name.

-u uses the time of the last access instead of the last modification for sorting (-t) and/or display (-l).

-1 forces one entry per line of output format; this is the default when output is not to a terminal.

-A is the same as **-a** except that it does not display dot and dot-dot.

-C forces multi-column output; this is the default when output is to a terminal.

-L lists files or directories referenced by symbolic links, rather than the symbolic links themselves.

-F uses single trailing characters to indicate entry types:

/	directory
=	socket
@	symbolic link
*	executable file

-R recursively lists subdirectories encountered.

name specifies the name of a file or a directory. The arguments are first sorted appropriately, but file arguments are processed before directories and their contents.

DESCRIPTION

The mode displayed under the -l option contains 11 characters. The first character indicates the type of entry:

d	directory;
b	block-type special file;
c	character-type special file;
l	symbolic link;
s	socket, or
-	plain file.

The next nine characters are interpreted as three sets of three bits each. The first set refers to owner permissions; the next refers to permissions to others in the same user-group; and the last to permissions to all others.

Within each set the three characters indicate permission respectively to read, to write, or to execute the file as a program. For a directory, 'execute' permission is interpreted to mean permission to search the directory. The permissions are indicated as follows:

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

The group-execute permission character is given as s if the file has the set-group-id bit set; likewise the user-execute permission character is given as s if the file has the set-user-id bit set.

The last character of the mode (normally x or hyphen) is t if the 1000 bit of the mode is on. See chmod for the meaning of this mode.

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

FILES

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

NOTES

- Newline and tab are considered printing characters in file names.
- The output device is assumed to be 80 columns wide.
- The option setting based on whether the output is a teletype is undesirable as ls -s is much different than ls -s | lpr. On the other hand, not doing this setting would break old shell scripts that use ls.

m4

macro processor

The m4 command invokes a macro processor intended as a front end for Ratfor, C, and other languages. Each specified file is processed in order and the result written on the standard output.

---

m4 [*file...*]

ARGUMENTS

file specifies an input file. A hyphen specifies the standard input (the default).

DESCRIPTION

Macro calls have the form

name(arg1,arg2, ... , argn)

The left parenthesis must immediately follow the name of the macro. If a defined macro name is not followed by a left parenthesis, it is deemed to have no arguments. Leading unquoted blanks, tabs, and newlines are ignored while collecting arguments. Potential macro names consist of alphabetic letters, digits, and underscore (_), where the first character is not a digit.

Left and right single quotes are used to delimit strings. The value of a quoted string is the string stripped of the quotes.

When a macro name is recognized, its arguments are collected by searching for a matching right parenthesis. Macro evaluation proceeds normally during the collection of the arguments, and any commas or right parentheses that happen to turn up within the value of a nested call are as effective as those in the original input text. After argument collection, the value of the macro is pushed back onto the input stream and rescanned.

The m4 command makes available the following built-in macros. They can be redefined, but once this is done the original meaning is lost. Their values are null unless otherwise stated.

define installs the second argument as the value of the macro whose name is the first argument. Each occurrence of $*n* in the replacement text, where *n* is a digit, is replaced by the *n*-th argument. Argument 0 is the name of the macro; missing arguments are replaced by the null string.

undefine removes the definition of the macro named in its argument.

ifdef if the first argument is defined, the value is the second argument, otherwise the third. If there is no third argument, the value is null. The word *unix* is predefined on UNIX versions of m4.

changequote changes quote characters to the first and second arguments. *Changequote* without arguments restores the original values.

divert changes the current output stream to its (digit-string) argument. *M4* maintains ten output streams, numbered 0–9. The final output is the concatenation of the streams in numerical order; initially stream 0 is the current stream. Output diverted to a stream other than 0 through 9 is discarded.

undivert causes immediate output of text from diversions named as arguments, or all diversions if no argument. Text can be undiverted into another diversion. Undiverting discards the diverted text.

divnum returns the value of the current output stream.

dnl reads and discards characters up to and including the next newline.

ifelse has three or more arguments. If the first argument is the same string as the second, the value is the third argument. If not, and if there are more than four arguments, the process is repeated with arguments 4, 5, 6 and 7. Otherwise, the value is either the fourth string, or, if it is not present, null.

incr returns the value of its argument incremented by 1. The value of the argument is calculated by interpreting an initial digit-string as a decimal number.

eval evaluates its argument as an arithmetic expression, using 32-bit arithmetic. Operators include plus sign, hyphen, asterisk, slash, per cent, and circumflex (exponentiation); relationals; parentheses.

len returns the number of characters in its argument.

index returns the position in its first argument where the second argument begins (zero origin), or −1 if the second argument does not occur.

substr returns a substring of its first argument. The second argument is a zero origin number selecting the first character; the third argument indicates the length of the substring. A missing third argument is taken to be large enough to extend to the end of the first string.

translit transliterates the characters in its first argument from the set given by the second argument to the set given by the third. No abbreviations are permitted.

include returns the contents of the file named in the argument.

sinclude is identical to *include*, except that it says nothing if the file is inaccessible.

syscmd executes the command given in the first argument. No value is returned.

maketemp fills in a string of XXXXX in its argument with the current process id.

errprint prints its argument on the diagnostic output file.

dumpdef prints current names and definitions, for the named items, or for all if no arguments are given.

SEE ALSO

Concentrix-2800 Programming Tools Guide: The M4 Macro Processor

machid fx2800

get machine id

The machid command prints the machine identification of the current machine.

/usr/alliant/machid

DESCRIPTION

On any FX/2800 family computer (FX/2800, FX/800, SRM/1) the identifying string is fx2800.

A link to machid is also provided, fx2800. If machid is invoked through that link, nothing is printed but a 0 exit status is returned. If the user wishes to make other computer types a link to machid, a 1 exit status will be returned if invoked through one of those alternative links. This facility is intended to assist in writing portable shell scripts and makefiles.

To make additional links, use the following procedure:

```
ln -s /usr/alliant/machid /usr/alliant/<machinetype>
```

mail

send and receive mail

The mail command invokes an intelligent mail processing system that has a command syntax reminiscent of ed with lines replaced by messages.

---

mail [*option...*] [**-s** *subject*] [*person...*]

mail [*option...*] [**-f** [*name*]]

mail [*option...*] **-f** *user*

ARGUMENTS

-d turns on debugging information.

-v displays details of delivery. Use when invoking sendmail. Also can be enabled by setting the option *verbose*.

-i ignores tty interrupt signals (particularly useful when using mail on noisy phone lines that generate spurious CTRL C or DELETE characters).

-n suppresses the reading of /usr/lib/Mail.rc.

-N suppresses the initial printing of headers.

-s *subject* specifies a subject (be careful to quote *subject*s containing spaces).

person is normally a user name recognized by login. To denote a recipient on a remote system, prefix *person* by the system name and exclamation mark (see the section below on mail addressing and the man page on uucp).

-f *name* reads the contents of the specified file (default is mbox) for processing; when you quit, mail writes undeleted messages back to this file.

-u *user* reads a specified user's mail instead of your own (equivalent to: mail -f /usr/spool/mail/user).

The following command line options are also recognized, but are intended for use by programs invoking mail and not for people. The -h and -r options are for network mail forwarding and are not used in practice since mail forwarding is now handled separately.

-T *file* arranges to print on *file* the contents of the *article-id* fields of all messages that were either read or deleted. -T is for the readnews program and should NOT be used for reading your mail.

-h *number* passes on hop count information. Mail takes the number, increments it, and passes it with -h to the mail delivery system. -h only has effect when sending mail and is used for network mail forwarding.

-r *name* used for network mail forwarding: interprets *name* as the sender of the message. The *name* and -r are simply sent along to the mail delivery system. Also, mail waits for the message to be sent and returns the exit status. Also restricts formatting of message.

SUMMARY OF COMMANDS

!	Single command escape to shell
-	Back up to previous message
alias	Define an alias as a set of user names
alternates	List other names you are known by
chdir	Change working directory, home by default
copy	Copy a message to a file or folder
delete	Delete a list of messages
dt (also dp)	Delete current message, type next message
edit	Edit a list of messages
else	Start of else part of conditional; see if
endif	End of conditional statement; see if
exit	Leave mail without changing anything
file	Interrogate/change current mail file
folder	Same as file
folders	List the folders in your folder directory
from	List headers of a list of messages
headers	List current window of messages
help	Print brief summary of *mail* commands
hold	Same as preserve
if	Conditional execution of *mail* commands
ignore	(ignore has been superseded by retain) Set/examine list of ignored header fields
list	List valid mail commands
local	List other names for the local host
mail	Send mail to specified names
mbox	Arrange to save a list of messages in *mbox*
next	Go to next message and type it
preserve	Arrange to leave list of messages in system mailbox
Print	Type message with ignored fields
quit	Leave mail; update system mailbox, *mbox* as appropriate
reply	Compose a reply to a message
Reply	Reply to author of message only
retain	Add the list of header fields named to the "retain list". Only the header fields in the retain list are shown on your terminal when you print a message. If retain is used with no arguments, it shows the retain list.
save	Append messages, headers included, on a file
set	Set binary or valued options
shell	Invoke an interactive shell
source	Read mail commands from a file.
top	Print first so many (five by default) lines of list of messages
type	Print messages
Type	Type message with ignored fields
undelete	Undelete list of messages
unread	Marks a list of messages as having not been read
unset	Undo the operation of a set
visual	Invoke visual editor on a list of messages
write	Append messages to a file, don't include headers
xit	Leave mail without changing anything.
z	Scroll to next/previous screen-full of headers

SUMMARY OF OPTIONS (VARIABLES)

These options are set using the **set** and **unset** commands.

EDITOR	*valued*	Pathname of editor for ~e and edit
PAGER	*valued*	Pathname of the program to use in the *more* command or when the *crt* variable is set
Replyall	*binary*	Reverses the sense of the *reply* and *Reply* commands
SHELL	*valued*	Pathname of shell for shell, ~! and !
VISUAL	*valued*	Pathname of screen editor for ~v, visual
append	*binary*	Always append messages to end of *mbox*
ask	*binary*	Prompt user for Subject: field when sending
askcc	*binary*	Prompt user for additional Cc's at end of message
autoprint	*binary*	Print next message after delete
crt	*valued*	Minimum number of lines before using the program defined by *PAGER*
debug	*binary*	Print out debugging information
dot	*binary*	Accept . alone on line to terminate message input
escape	*valued*	Escape character to be used instead of ~
folder	*valued*	Directory to store folders in
hold	*binary*	Hold messages in system mailbox by default
ignore	*binary*	Ignore CTRL C while sending mail
ignoreeof	*binary*	Don't terminate letters/command input with CTRL D
keep	*binary*	Don't unlink system mailbox when empty
keepsave	*binary*	Don't delete saved messages by default
metoo	*binary*	Include sending user in aliases
noheader	*binary*	Suppress initial printing of version and headers
nosave	*binary*	Don't save partial letter in *dead.letter*
quiet	*binary*	Suppress printing of *mail* version and message numbers
record	*valued*	File to save all outgoing mail in
screen	*valued*	Size of window of message headers for z, etc.
sendmail	*valued*	Choose alternate mail delivery system
toplines	*valued*	Number of lines to print in top
verbose	*binary*	Invoke sendmail with the –v flag

SUMMARY OF ESCAPES

~! *command*	Execute shell command
~b	Add names to blind carbon list
~c *name ...*	Add names to Cc: field
~d	Read *dead.letter* into message
~e	Invoke text editor on partial message
~f *messages*	Read named messages
~h	Edit the header fields
~m *messages*	Read named messages, right shift by tab
~p	Print message entered so far
~q	Abort entry of letter; like CTRL C
~r *filename*	Read file into message
~s *string*	Set Subject: field to *string*
~t *name ...*	Add names to To: field
~v	Invoke screen editor on message
~w *filename*	Write message on file
~| *command*	Pipe message through *command*
~~ *string*	Quote a ~ in front of *string*

MAIL ADDRESSING

Mail addresses are based on the ARPANET protocol listed at the end of this manual page. These addresses are in the general format

```
user@domain
```

where a domain is a hierarchical dot-separated list of subdomains. For example, the address

```
eric@monet.berkeley.edu
```

is normally interpreted from right to left: the message should go to the ARPA name tables (which do not correspond exactly to the physical ARPANET), then to the Berkeley gateway, after which it should go to the local host monet. When the message reaches monet it is delivered to the user eric.

Unlike some other forms of addressing, this does not imply any routing. Thus, although this address is specified as an ARPA address, it might travel by an alternate route if that were more convenient or efficient. For example, at Berkeley, the associated message would probably go directly to monet over the Ethernet rather than going via the Berkeley ARPANET gateway.

Under certain circumstances it may not be necessary to type the entire domain name. In general, anything following the first dot may be omitted if it is the same as the domain from which you are sending the message. For example, a user on "calder.berkeley.edu" could send to "eric@monet" without adding the "berkeley.edu" since it is the same on both sending and receiving hosts.

Certain other abbreviations may be permitted as special cases. For example, at Berkeley, ARPANET hosts may be referenced without adding the "berkeley.edu" as long as their names do not conflict with a local host name.

Certain old address formats are converted to the new format to provide compatibility with the previous mail system. In particular,

```
user@host.ARPA
```

is allowed and

```
host:user
```

is converted to

```
host@user
```

to be consistent with the rcp command.

Also, the syntax

```
host!user
```

is converted to:

```
user@host.UUCP
```

This is normally converted back to the "host!user" form before being sent on for compatibility with older UUCP hosts.

The current implementation is not able to route messages automatically through the UUCP network. Until that time you must explicitly tell the mail system which hosts to send your message through to get to your final destination.

Domain names (i.e., anything after the "@" sign) may be given in any mixture of upper and lower case with the exception of UUCP hostnames. Most hosts accept any combination of case in user names, with the notable exception of MULTICS sites.

Under some circumstances it may be necessary to route a message through several hosts to get it to the final destination. Normally this routing is done automatically, but sometimes it is desirable to route the message manually. Addresses which show these relays are termed "route– addrs." These use the syntax:

```
<@hosta,@hostb:user@hostc>
```

This specifies that the message should be sent to hosta, from there to hostb, and finally to hostc. This path is forced even if there is a more efficient path to hostc.

Route-addrs occur frequently on return addresses, since these are generally augmented by the software at each host. It is generally possible to ignore all but the "user@domain" part of the address to determine the actual sender.

Every site is required to have a user or user alias designated "postmaster" to which problems with the mail system may be addressed.

Some other networks can be reached by giving the name of the network as the last component of the domain. This is not a standard feature and may not be supported at all sites. For example, messages to CSNET or BITNET sites can often be sent to "user@host.CSNET" or "user@host.BITNET" respectively.

Notes

- The RFC822 group syntax ("group:user1,user2,user3;") is not supported except in the special case of "group:;" because of a conflict with old berknet–style addresses.
- Route-Address syntax is grotty.
- UUCP- and ARPANET-style addresses do not coexist politely.

FILES

/usr/spool/mail/*	post office
~/mbox	your old mail
~/.mailrc	file giving initial mail commands
/tmp/R#	temporary for editor escape
/usr/lib/Mail.help*	help files
/usr/lib/Mail.rc	system initialization file
Message*	temporary for editing messages

SEE ALSO

binmail, fmt
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: aliases
Concentrix-2800 System Administration: newaliases, sendmail
Concentrix-2800 System User's Guide: Mail Reference
Crocker, D. H., *Standard for the Format of Arpa Internet Text Messages*, RFC822.

make GNU_make

maintain program groups

The GNU_make command executes a file of commands to update one or more target files. Make updates a target if it depends on prerequisite files that have been modified since the target was last modified, or if the target does not exist.

---

make [**-f** *makefile*] [*option...*] *target...*

ARGUMENTS

-b -m these options are ignored for compatibility with other versions of make.

-C *dir* changes to directory *dir* before reading the makefiles or doing anything else. If multiple -C options are specified, each is interpreted relative to the previous one: -C / -C etc is equivalent to -C /etc. This is typically used with recursive invocations of make.

-d prints debugging information in addition to normal processing. The debugging information says which files are being considered for remaking, which file-times are being compared and with what results, which files actually need to be remade, which implicit rules are considered and which are applied – everything interesting about how make decides what to do.

-e gives variables taken from the environment precedence over variables from makefiles.

-f *makefile* by default, make tries GNUmakefile, makefile and Makefile in order. You can supply more than one -f option. A hyphen in place of *makefile* specifies the standard input.

-I *dir* specifies a directory *dir* to search for included makefiles. If several -I options are used to specify several directories, the directories are searched in the order specified. Unlike the arguments to other flags of make, directories given with -I flags may come directly after the flag: -I*dir* is allowed, as well as -I *dir*. This syntax is allowed for compatibility with the C preprocessor's -I flag.

-i ignores all errors in commands executed to remake files. This is equivalent to the special entry .IGNORE:.

-j*n* tells make to execute in parallel when possible, with *n* as the maximum number of parallel jobs. *n* defaults to 1, and can be as high as the number of CEs plus 1. If more than one -j option exists, the last one is effective.

-k abandons work on the current entry when a command returns nonzero status, but continues on branches that do not depend on the current entry.

-l *load* specifies that no new jobs (commands) should be started if there are other jobs running and the load average is at least *load* (a floating-point number). With no argument, removes a previous load limit.

-n traces and displays, but does not execute, the commands needed to update the targets.

-p prints the data base (rules and variable values) that result from reading the makefiles; then executes as usual or as otherwise specified. This also prints the version information given by the -v switch (see below). To print the data base without trying to remake any files, use the following command:

```
make -p -f /dev/null
```

-q "questions" silently whether the targets are up to date already, but executes no commands in either case. Returns a zero exit status if the specified targets are already up to date, nonzero otherwise.

-r eliminates use of the built-in implicit rules. Also clears out the default list of suffixes for suffix rules.This is equivalent to an initial special entry `.SUFFIXES:` with no list.

-s is equivalent to the special entry `.SILENT:`.

-S cancels the effect of the -k option. This is never necessary except in a recursive make where -k might be inherited from the top-level make via MAKEFLAGS or if you set -k in MAKEFLAGS in your environment.

-t touches (updates) the modified date of targets, without executing any commands.

-v prints the version of the make program plus a copyright, a list of authors and a notice that there is no warranty. After this information is printed, processing continues normally. To get this information without doing anything else, use make -v -f /dev/null.

-w prints a message containing the working directory before and after executing the makefile. This may be useful for tracking down errors from complicated nests of recursive make commands.

-W *file* "what if" pretends that the target *file* has just been modified.The given files' modification times are recorded by make as being the present time, although the actual modification times remain the same. When used in conjunction with the -n flag, the -W flag provides a way to see what would happen if you were to modify specific files.

target is typically a program but can be a document, a data file, and so forth.

DESCRIPTION

The make utility determines automatically which pieces of a large program need to be recompiled, and issues the commands to recompile them. This man page briefly describes the GNU implementation of make, which was written by Richard Stallman and Roland McGrath. For complete information on the many features of GNU make, see the *Concentrix-2800 Programming Tools Guide* chapter on this utility.

To prepare to use make you must write a file called the *makefile* (defaults to GNUmakefile, makefile, or Makefile, in that order, in the current working directory) that describes the relationships among files in your program, and then states the commands for updating each file. In a program, typically the executable file is updated from object files, which are in turn made by compiling source files.

Once a suitable makefile exists, each time you change some source files, this simple shell command:

```
make
```

suffices to perform all necessary recompilations. The make program uses the makefile data base and the last-modification times of the files to decide which of the files need to be updated. For each of those files, it issues the commands recorded in the data base.

The command arguments to make are used to control which files to recompile, or how.

Makefile contains a sequence of entries that specify dependencies. The first line of an entry is a blank-separated list of targets, then a colon, then a list of prerequisite files. Text following a semicolon, and all following lines that begin with a tab, are shell commands to be executed to update the target.

If a name appears on the left of more than one colon line, it depends on all of the names on the right of the colon on those lines, but only one command sequence can be specified for it. If a name appears on a line with a double colon (::), the command sequence following that line is performed only if the name is out of date with respect to the names to the right of the double colon, and is not affected by other double colon lines on which that name may appear.

Sharp and newline surround comments.

The following makefile says that pgm depends on two files a.o and b.o, and that they in turn depend on .c files and a common file incl.

```
pgm: a.o b.o
        cc a.o b.o -lm -o pgm
a.o: incl a.c
        cc -c a.c
b.o: incl b.c
        cc -c b.c
```

Makefile entries of the form

```
string1 = string2
```

are macro definitions. Subsequent appearances of $(*string1*) or ${*string1*} are replaced by *string2*. If *string1* is a single character, the parentheses or braces are optional.

Make infers prerequisites for files for which *makefile* gives no construction commands. For example, a .c file can be inferred as prerequisite for a .o file and be compiled to produce the .o file. Thus the preceding example can be done more briefly:

```
pgm: a.o b.o
        cc a.o b.o -lm -o pgm
a.o b.o: incl
```

Prerequisites are inferred according to selected suffixes listed as the prerequisites for the special name .SUFFIXES; multiple lists accumulate; an empty list clears what came before. Order is significant; the first possible name for which both a file and a rule as described in the next paragraph exist is inferred. The default list is

```
.SUFFIXES: .out .o .c .e .r .f .y .l .s .p
```

The rule to create a file with suffix *s2* that depends on a similarly named file with suffix *s1* is specified as an entry for the "target" *s1s2*. In such an entry, the special macro $* stands for the target name with suffix deleted, $@ for the full target name, $< for the complete list of prerequisites, and $? for the list of prerequisites that are out of date. For example, a rule for making optimized .o files from .c files is

```
.c.o: ; cc -c -O -o $@ $*.c
```

Certain macros are used by the default inference rules to communicate optional arguments to any resulting compilations. In particular, CFLAGS is used for cc options, FFLAGS for fortran options, PFLAGS for pc options, and LFLAGS and YFLAGS for lex and yacc options. In addition, the macro MFLAGS is filled in with the initial command line options supplied to make. This simplifies maintaining a hierarchy of makefiles as one can then invoke make on makefiles in subdirectories and pass along useful options such as -k.

Another special macro is VPATH. The VPATH macro should be set to a list of directories separated by colons. When make searches for a file as a result of a dependency relation, it first searches the current directory and then each of the directories on the VPATH list. If the file is found, the actual path to the file is used, rather than just the filename. If VPATH is not defined, then only the current directory is searched.

One use for VPATH is when one has several programs that compile from the same source. The source can be kept in one directory and each set of object files (along with a separate makefile) would be in a separate subdirectory. The VPATH macro would point to the source directory in this case.

Command lines are executed one at a time, each by its own shell. A line is displayed when it is executed unless the special target .SILENT is in *makefile*, or the first character of the command is an at sign (@).

Commands returning nonzero status cause make to terminate unless the special target .IGNORE is in *makefile* or the command begins with tab, hyphen.

Interrupt and quit cause the target to be deleted unless the target is a directory or depends on the special name .PRECIOUS.

The dependencies of the special target .PHONY are considered to be phony targets. Using this special built-in name as the target causes make to run its commands unconditionally, regardless of whether a file of that name exists, or what its last modification time is.

NOTES

- Some commands return nonzero status inappropriately. Use -i to overcome the difficulty.
- Commands that are directly executed by the shell, notably cd, are ineffectual across newlines in make.
- VPATH is intended to function like the System V VPATH. but there is no guarantee that it functions identically.

FILES

GNUmakefile
makefile
Makefile

SEE ALSO

touch
Concentrix-2800 Programming Tools Guide: GNUMake – A Program for Maintaining Computer Programs

make

maintain program groups (old)

The make command executes a file of commands to update one or more target files. Make updates a target if it depends on prerequisite files that have been modified since the target was last modified, or if the target does not exist. This (BSD) version has been replaced with the GNU_make command.

/usr/old/make [**-f** *makefile*] [*option...*] *file...*

ARGUMENTS

-f*makefile* by default, *make* tries makefile and Makefile in order. You can supply more than one -f option. A hyphen specifies the standard input (the default).

-i is equivalent to the special entry .IGNORE:.

-j*n* tells make to execute in parallel when possible, with *n* as the maximum number of parallel jobs. *n* defaults to 1, and can be as high as the number of CEs plus 1.

-k abandons work on the current entry when a command returns nonzero status, but continues on branches that do not depend on the current entry.

-n traces and displays, but does not execute, the commands needed to update the targets.

-t touches (updates) the modified date of targets, without executing any commands.

-r is equivalent to an initial special entry .SUFFIXES: with no list.

-s is equivalent to the special entry .SILENT:.

file is typically a program but can be a document, a data file, and so forth.

DESCRIPTION

Makefile contains a sequence of entries that specify dependencies. The first line of an entry is a blank-separated list of targets, then a colon, then a list of prerequisite files. Text following a semicolon, and all following lines that begin with a tab, are shell commands to be executed to update the target.

If a name appears on the left of more than one colon line, it depends on all of the names on the right of the colon on those lines, but only one command sequence can be specified for it. If a name appears on a line with a double colon (::), the command sequence following that line is performed only if the name is out of date with respect to

the names to the right of the double colon, and is not affected by other double colon lines on which that name may appear.

Sharp and newline surround comments.

The following makefile says that pgm depends on two files a.o and b.o, and that they in turn depend on .c files and a common file incl.

```
pgm: a.o b.o
        cc a.o b.o -lm -o pgm
a.o: incl a.c
        cc -c a.c
b.o: incl b.c
        cc -c b.c
```

Makefile entries of the form

```
string1 = string2
```

are macro definitions. Subsequent appearances of $(*string1*) or ${*string1*} are replaced by *string2*. If *string1* is a single character, the parentheses or braces are optional.

Make infers prerequisites for files for which *makefile* gives no construction commands. For example, a .c file can be inferred as prerequisite for a .o file and be compiled to produce the .o file. Thus the preceding example can be done more briefly:

```
pgm: a.o b.o
        cc a.o b.o -lm -o pgm
a.o b.o: incl
```

Prerequisites are inferred according to selected suffixes listed as the prerequisites for the special name .SUFFIXES; multiple lists accumulate; an empty list clears what came before. Order is significant; the first possible name for which both a file and a rule as described in the next rpara exist is inferred. The default list is

```
.SUFFIXES: .out .o .c .e .r .f .y .l .s .p
```

The rule to create a file with suffix *s2* that depends on a similarly named file with suffix *s1* is specified as an entry for the "target" *s1s2*. In such an entry, the special macro $* stands for the target name with suffix deleted, $@ for the full target name, $< for the complete list of prerequisites, and $? for the list of prerequisites that are out of date. For example, a rule for making optimized .o files from .c files is

```
.c.o: ; cc -c -O -o $@ $*.c
```

Certain macros are used by the default inference rules to communicate optional arguments to any resulting compilations. In particular, CFLAGS is used for cc options, FFLAGS for fortran options, PFLAGS for pc options, and LFLAGS and YFLAGS for lex and yacc options. In addition, the macro MFLAGS is filled in with the initial command line options supplied to make. This simplifies maintaining a hierarchy of makefiles as one can then invoke make on makefiles in subdirectories and pass along useful options such as -k.

Another special macro is VPATH. The VPATH macro should be set to a list of directories separated by colons. When make searches for a file as a result of a dependency relation, it will first search the current directory and then each of the directories on the VPATH list. If the file is found, the actual path to the file will be used, rather than just the filename. If VPATH is not defined, then only the current directory is searched.

One use for VPATH is when one has several programs that compile from the same source. The source can be kept in one directory and each set of object files (along with a separate makefile) would be in a separate subdirectory. The VPATH macro would point to the source directory in this case.

Command lines are executed one at a time, each by its own shell. A line is displayed when it is executed unless the special target .SILENT is in *makefile*, or the first character of the command is an at sign (@).

Commands returning nonzero status cause make to terminate unless the special target .IGNORE is in *makefile* or the command begins with tab, hyphen.

Interrupt and quit cause the target to be deleted unless the target is a directory or depends on the special name .PRECIOUS.

NOTES

- Some commands return nonzero status inappropriately. Use –i to overcome the difficulty.
- Commands that are directly executed by the shell, notably cd, are ineffectual across newlines in make.
- VPATH is intended to function like the System V VPATH. but there is no guarantee that it functions identically.

FILES

makefile
Makefile

SEE ALSO

touch
Concentrix Programming Tools Guide: Make – A Program for Maintaining Computer Programs

man

read on-line documentation

The man command finds and displays information in the on-line documentation set. When the standard output is a terminal, man pipes its output through more -s for convenience.

---

man [-] [-a] [-M *path*[:*path*]...] [[-s] *section*] *title...*

man [-M *path*[:*path*]...] -f *title...* | **whatis** *title...*

man [-M *path*[:*path*]...] -k *keyword...* | **apropos** *keyword...*

ARGUMENTS

-	pipes the output through cat -s.
-a	displays the 1-line entry for the specified item from the whatis database before displaying the full-page entry from the man database. If the item is not what you want, you can quit and re-enter.
-t	sends the output to troff. (Obsolete, but supported for backward compatibility.)
-M *dir*	specifies directory(s) in which to search. This option is intended for debugging new man files. Separate directories in the list with a colon. The search path can also be set with the MANPATH environment variable. By default, man searches in /usr/man, the standard location for on-line documentation.
-p *dir*	equivalent to -M. (Obsolete, but supported for backward compatibility.)
-s *section*	specifies a "section" (explained below) in which to search. If no *section* is specified, man searches through all of the sections. The system checks the environmental variable MANSECT which is prepended to the list of sections searched by default. This enables searching 2 and 3 before a and f for example.
title	searches for documentation files associated with the specified *title*.
-f *title*	displays a one-line synopsis of each on-line documentation file whose name matches the specified *title* (equivalent to the whatis command). When using -f, man does not send output through more -s.
-k *keyword*	displays a one-line synopsis of each on-line documentation file whose name or description contains the specified *keyword* (equivalent to the

apropos command). When using -k, man does not send output through more -s.

DESCRIPTION

When searching for documentation files associated with the specified *title*, man checks for the existence of files in nroff source format and nroff output format.

If only the output file exists, man displays it on the terminal.

If only the source file exists, man invokes nroff and attempts to create a new output file, and then to display the file on the terminal. If an error occurs (cat? directory does not exist, no write access, etc.) man invokes nroff and sends the output directly to the terminal.

If both files exist, man compares their creation dates. If the source file is newer, *man* behaves as if only the source file exists. If the output file is newer, man displays it on the terminal.

Closely related to the man command is the whatisin command. Whatisin shows you a summary of what is in each section of the man database.

SECTION NAMES

The Alliant documentation set closely follows the UNIX documentation set. The relationship between the two documentation sets is summarized below.

Alliant Section	Meaning
0	Overview of features and documentation
1	Commands and utilities
1b	Bourne shell
1c	C shell
1o	old commands
1u	unsupported commands
2	System calls
3	C library routines
3r	C library Remote Procedure Call routines
4	Special files
5	File formats
6	Games
7	Miscellaneous
8	System administration commands
a	Alliant libraries
ac	FX/ and FX/2800 Series common library
am	FX/ and FX/2800 Series signal processing library
am	FX/ and FX/2800 Series linear algebra library
am	FX/ and FX/2800 Series cray library
c	FX/ and FX/2800 C Programmer's Handbook
f	FX/ and FX/2800 Fortran Language Manual
ff	FX/ and FX/2800 Fortran: format specifications
fi	FX/ and FX/2800 Fortran: intrinsics
fl	FX/ and FX/2800 Fortran: library routines
fs	FX/ and FX/2800 Fortran: statements
r	FX/ and FX/2800 Series PHIGS/PHIGS+ libraries

Alliant Section	Alliant Manual
0	none
1	*Concentrix-2800 Commands and Applications Manual*
1b	*Concentrix-2800 Commands and Applications Manual* and *Concentrix-2800 System User's Guide*
1c	*Concentrix-2800 Commands and Applications Manual* and *Concentrix-2800 System User's Guide*
1o	none
1u	none
2	*Concentrix-2800 System Reference Manual: System Calls and C Library Routines*
3	*Concentrix-2800 System Reference Manual: System Calls and C Library Routines*
3r	*Concentrix-2800 System Reference Manual: System Calls and C Library Routines*
4	*Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files and File Formats*
5	*Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files and File Formats*
6	none
7	none
8	*Concentrix-2800* and *Concentrix-2800-2800 System Administration Manual* a *Alliant Libraries*
ac	*FX/* and *FX/2800 Series Common Library*
am	*FX/* and *FX/2800 Series Signal Processing Library*
am	*FX/* and *FX/2800 Series Linear Algebra Library*
am	*FX/* and *FX/2800 Series Cray Library*
c	*FX/* and *FX/2800 C Programmer's Handbook*
f	*FX/* and *FX/2800 Fortran Language Manual*
ff	*FX/* and *FX/2800 Fortran Language Manual*: *format specifications*
fi	*FX/* and *FX/2800 Fortran Language Manual*: *intrinsics*
fl	*FX/* and *FX/2800 Fortran Language Manual: library routines*
fs	*FX/* and *FX/2800 Fortran Lnaguage Manual: statements*
rc	*Alliant PHIGS/PHIGS+ C Subroutine Reference*
rf	*Alliant PHIGS/PHIGS+ Fortran Subroutine Reference*

Alliant Section	UNIX Manual
0	n/a
1	Commands
1b	Commands
1c	Commands
1o	n/a
1u	n/a
2	System calls
3	Subroutines
3r	n/a
4	Special files
5	File formats
6	Games
7	Miscellaneous
8	Maintenance

a	n/a
ac	n/a
am	n/a
f	n/a
ff	n/a
fi	n/a
fl	n/a
fs	n/a
r	n/a

The man database also contains the sections "new", "old", "local" and "public". You may enter one of these strings in place of a section number.

NOTES

- By eliminating /usr/man/cat?, it is possible to save approximately 3MB of disk space at the expense of interactive performance. You can recreate /usr/man/cat? with catman (see *Concentrix-2800 System Administration*).

DIAGNOSTICS

- If you use the -M option, and name a directory that does not exist, the error message is somewhat misleading. Suppose the directory /usr/foo does not exist. If you type

  ```
  man -M /usr/foo ls
  ```

  you get the error message "`No manual entry for ls.`" You should get an error message indicating that the directory /usr/foo does not exist.

FILES

/usr/man/man?/*	on-line documentation in nroff source format
/usr/man/cat?/*	on-line documentation in nroff output format
/usr/man/whatis	on-line documentation database file

SEE ALSO

apropos, cat, info, more, nroff, troff, whatis, whatisin, whereis
Concentrix-2800 System Administration: catman

mesg

permit or deny messages

The mesg command controls the appearance of messages on your terminal via write and talk by setting non-user write permission.

---

mesg [yn]

ARGUMENTS

y enables messages.

n disables messages.

With no argument, *mesg* reports the current state.

DIAGNOSTICS

- Exit status is 0 if messages are receivable, 1 if not, 2 on error.

FILES

/dev/tty*

SEE ALSO

write, talk

mkdir

create directory

The mkdir command creates the specified directories according to your file creation mask (see umask). The standard entries dot (.) for the directory itself, and dot-dot (..) for its parent, are created automatically.

---

mkdir *dirname...*

ARGUMENTS

dirname specifies the name of the directory to create.

DESCRIPTION

Mkdir requires write permission in the parent directory.

SEE ALSO

rmdir

mkstr

create error message file from C source

The mkstr command extracts error messages from C source code and creates separate error message and program files. It allows you to reduce the size of programs with large numbers of error messages and to reduce run-time overhead as error messages do not have to be constantly swapped in and out.

---

mkstr [-] *messagefile prefix file...*

ARGUMENTS

-	places error messages at the end of the specified message file.
messagefile	specifies the name of a file of error messages.
prefix	specifies a prefix for each *file*.
file	specifies the name of a C source program file.

DESCRIPTION

Mkstr reads each of the specified *files*, placing a processed version of the input file in a file whose name consists of the specified *prefix* and the original name. A typical usage of mkstr would be

```
mkstr pistrings xx *.c
```

This command would cause all the error messages from the C source files in the current directory to be placed in the file *pistrings* and processed copies of the source for these files to be placed in files whose names are prefixed with *xx*.

To extract error messages, mkstr keys on the string `error("` in the input stream. Each time it occurs, the C string starting at the double quote is placed in the message file followed by a null character and a new-line character; the null character terminates the message so it can be easily used when retrieved, the new-line character makes it possible to *cat* the error message file to see its contents. The processed copy of the input file then contains a *lseek* pointer into the file that can be used to retrieve the message.

EXAMPLE

```
char    efilname[] = "/usr/lib/pi_strings";
int     efil = -1;

error(a1, a2, a3, a4)
{
        char buf[256];

        if (efil < 0) {
                efil = open(efilname, 0);
                if (efil < 0) {
oops:
                perror(efilname);
                exit(1);
                }
        }
        if (lseek(efil, (long) a1, 0) || read(efil, buf, 256) <= 0)
                goto oops;
        printf(buf, a2, a3, a4);
}
```

SEE ALSO

xstr
Concentrix-2800 System Reference Manual: System Calls and C Library Routines:
lseek

mon

monitor process/system status (crt display)

The mon command monitors the utilization of system resources and displays status information in real time. Mon has three modes of operation: family mode, system mode, and computing resource mode, as described below. The default is family mode. To exit mon, press CTRL C.

---

mon [*user* | *pid*] [*option...*]

ARGUMENTS

user, pid specifies a target process to monitor in family mode. Ignored in system or computing resource mode.

If *user* is specified, mon displays a tree of processes whose root process is owned by *user*. If *user* owns multiple process trees, mon displays one, chosen at random.

If *pid* is specified, mon displays only the tree of processes whose root process is *pid*.

-a displays all processes owned by *user* in family mode. Ignored if -t option is used.

-c enters computing resource mode.

-f *field...* specifies an arbitrary list of *field* names that override the default selection (as explained below). Only the first four characters of a *field* name are parsed; case is ignored. The -f option must be the last option on the command line (only field names can follow). For example:

```
mon -t -r -f user pid cput rsz stat s comm
```

If no field names are specified after the -f option, mon prints an error message showing the complete list of valid fields.

-i*interval* specifies the number of seconds to sleep between samples. The default sample *interval* is five seconds.

-m enters system mode and sorts processes by the amount of physical memory currently used by the whole system(RSZ). This option does not give information for just one user.

-N (requires installation of NeWS, a separately orderable product) shows system activity graphically, using the Alliant architecture diagram, on a workstation also running NeWS. You must first set the environment variable NEWSSERVER as follows, where *host* is the host name of the workstation:

```
% setenv NEWSSERVER 'setnewshost host'
```

-r displays a bar graph showing each installed computing resource (including the complex) and the percent of time that each resource spends in each state (can be used in family mode or system mode).

-t enters system mode and sorts processes by the amount of cpu time currently used. The percentages are weighted by the relative speed of the given resource (1 CE = 8 IPs on the FX/80 Series) and the results used to sort the processes.

-v displays swap/page space usage; alters the column displays to swapslot size.

INTERACTIVE COMMANDS

You can change the mon display interactively while the program is running. Type one of the following single letters to change the display:

a same as **-a** on the command line.

c enters computing resource mode.

h (or ?) displays a help frame containing a short list of the valid interactive commands. When you are done reading the help, type any character to go back to your previous display.

i sets the update interval, in seconds. Mon displays a prompt at the bottom of the screen. Type in the new interval (number of seconds) and press RETURN.

CTRL **L** refreshes the screen (useful when the display has become corrupted by other output).

m enters system mode and sorts processes by the amount of physical memory currently used (RSZ).

q causes mon to exit. Typing CTRL C also exits but may leave the terminal in reverse video mode.

r displays a bar graph showing each installed computing resource (including the complex) and the percent of time that each resource spends in each state (can be used in family mode or system mode).

t enters system mode and sorts processes by the amount of cpu time currently used. The percentages are weighted by the relative speed of the given resource (1 CE = 8 IPs) and the results used to sort the processes.

u allows you to watch a set of related processes. Mon displays a prompt at the bottom of the screen. You can enter a username, in which case mon tracks processes of that uid, or you can type in a pid in which case it tracks that process and its children.

Family Mode

In family mode, mon monitors a target process and all of its descendent processes or all processes owned by a user. At the top of the screen, mon prints a line showing the owner of the target process and the word ALL if the a flag has been specified to show all of the user's processes. Directly below is a status line for each process monitored (the family tree of the target process is traversed and displayed in youngest-child-first order). By default, a process status line consists of the following fields (see Field Names below for explanations):

```
PID  COMMAND  STAT  C  P  IP  CE  CL  %CPU  S  PF
```

System Mode

In system mode, mon monitors all processes in the system. Depending on the option used, mon displays sorted status lines for those processes consuming the most cpu time (-t) or the most physical memory (-m). Only processes that exist for the entire sampling interval appear.

By default, a process status line for the -t option consists of the following fields (see Field Names below for explanations):

```
USER  PID  %IP  %CE  %CL  CPUTIME  RSZ  COMMAND
```

By default, a process status line for the -m option consists of the following fields (see Field Names below for explanations):

```
PID  C  SW  S  VSZ  RSZ  PF  PR  STAT  USER  COMMAND
```

Computing Resource Mode

In computing resource mode (-c option), mon displays a bar graph showing the state of each installed processor and the computational complex (if present). The initial display shows statistics from system boot time to the present. The numbers shown with each resource are the same as shown in the system status line (as described below). The bars show percent of time each processor spends in each state (user, system, interrupt, lock-wait, and idle).

The System Status Line

Mon displays a status line at the bottom of the screen that shows the host name, the static and dynamic configuration of the system, the memory usage, and the current time. Unless you use the -c option or the -r option*, it appears in the form:

```
alliant  5 IP 12%    4 CE(4) 24%    1 CL(4) 99%   28.1M ( 54%) 4:45:35pm
```

(If you use the -c option or the -r option, the system status line shows only the host name, memory usage, and current time.)

The numbers preceding IP, CE, and CL, are the *effective* number of installed processors. For CE and for CL with dynamic detached CEs, this number varies and is not necessarily an integer. The number in parentheses is the number of CEs in the static configuration. For example:

`1 CL(4) 99%`	indicates one complex available with four attached CEs.
`0.3 CL(4) 99%`	indicates effectively .3 complex available (one complex available 30% of the time).

`4 CE(4) 24%` indicates effectively four detached CEs available, of which four are statically detached (cannot participate in the complex).

`6.1 CE(3) 12%` indicates effectively 6.1 detached CEs available, of which three are statically detached (cannot participate in the complex).

To the right in all cases is the percent of available resource utilized for user, system, and interrupt time (excluding idle and lockwait time).

In the example, 28.1M signifies that 28.1 megabytes of memory are free. To the right of this number is the percent of user memory the free number represents.

(User memory = installed memory - system memory.)

Field Names

The -f option allows you to select an arbitrary set of fields from the following list. Field names with a percent sign (%) refer to a percentage of the last sample interval.

C	Scheduling resource class (see setsched, showsched)
CE	Total cpu time on computational elements
%CE	Computational element usage (percentage)
CL	Total cpu time on the computational complex
%CL	Computational complex usage (percentage)
COMMAND	Command line (or portion thereof)
CPUTIME	Cpu time used
%CPU	Cpu time used (percentage)
INBLK	Input blocks read
IP	Total cpu time on interactive processors
%IP	Interactive processor usage (percentage)
NI	Current nice level (see nice)
OUBLK	Output blocks written
P	Current priority
PC	user process program counter
PF	Page fault (out of memory)
PID	Process identification number
PPID	Parent process identification number
PR	Page reclaim (in memory)
PROC_ADD	Address of proc structure
RSZ	Resident set size (physical memory in use)
S	Currently swapped out (no header)
SCH	Scheduler: returns "ts" if process is running on the timesharing scheduler, returns realtime priority (1 to 100) if process is running in realtime
SLEEPPC	Program counter of sleeping process
STA	Process state (Run, Stp, Slp, Idl, Zmb) (no header)
SW	Number of times swapped
SYM_SLEEP	Kernel symbol name (routine name) where a sleeping process is sleeping
SYSTEM	System time for process
%SYS	System time for process (percentage)
TT	Tty process is using
UID	User identification number of owner

USER	Username of owner
USR	User time for process
%USR	User time for process (percentage)
VSZ	Virtual set size (virtual memory needed for process)
WCHANNEL	Event process is sleeping on

When selecting your list of fields, make sure that your selection fits on a single line; otherwise, undesired scrolling may occur.

NOTES

- Things can change while mon is running; the picture it gives is only a close approximation of reality.
- When monitoring a single user, if there are too many processes to fit on the screen, mon shows only the oldest processes.
- Mon uses the curses library for cursor movement, and thus may run poorly on some terminals.
- Mon uses the curses database to determine the number of lines on your screen, and attempts to use all of those lines. To take advantage of this, however, you must make sure that: 1) the environment variable TERM is set to the type of terminal you are using; 2) the termcap database contains an entry for your terminal, with the "li" field set correctly.

  For example, suppose you are using a Sun Windows display with 50 lines and 80 columns. You can create a termcap entry that says:

```
sun50x80|Sun Windows with 50 lines and 80 columns:\
        :li#50:co#80:tc=sun:
```

  You then set the environment variable TERM to "sun50x80", and mon will use a 50 line display.

- The data displayed can be significantly affected by the sample interval chosen. A longer interval tends to smooth the results and show the steady state of the system load. Processes created or killed during the sample period do not appear in the top-cpu (-t) display, thus, the processes displayed may not account for all of the cpu time available. A shorter interval gives an instantaneous impression but does not necessarily represent an average distribution of activity.
- If you specify a PID that corresponds to a thread_id, mon dsplays the thread identifier separately from the root thread, under the COMMAND field, in brackets.

FILES

/dev/mem	read access required
/dev/kmem	read access required

SEE ALSO

ps, setsched, showsched, w

more

file perusal filter for crt viewing

The more command displays a text file one screenfull at a time on a video terminal. It normally pauses after each screenfull, displaying --More-- at the bottom of the screen. Pressing RETURN displays one more line. Pressing the space bar displays another screenfull.

---

more [-cdflsu] [-*windowsize*] [+*linenumber*] [+/*pattern*] [*name...*]

ARGUMENTS

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

-d prompts with the message "Press space to continue, 'q' to quit" at the end of each screenfull. If the user enteres illegal input, more responds with "Press 'h' for instructions".

-f counts logical, rather than screen lines (long lines are not folded). This option is recommended if nroff output is being piped through ul, which can generate escape sequences containing characters that would ordinarily occupy screen positions, but that do not display as part of an escape sequence. That can cause more to fold lines erroneously because they appear to be longer than they actually are.

-l suppresses special treatment of formfeeds (CTRL L). By default, more pauses after any line that contains a formfeed, as if the end of a screenfull had been reached. If a file begins with a formfeed, the screen is cleared before the file is displayed.

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

-u suppresses processing of underlining. By default, if the terminal can perform underlining or has a stand-out mode, more outputs appropriate escape sequences for underlined information in the source file.

-windowsize
is an integer that specifies the size (in lines) of the window that more uses.

+linenumber
specifies the line number at which to start up.

+/pattern
causes more to start up two lines before the line containing the specified regular expression.

DESCRIPTION

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

More looks in the environment variable MORE to preset any options desired. For example, if you prefer to view files using the *-c* mode of operation, the *csh* command `setenv MORE -c` or the sh command sequence `MORE='-c' ; export MORE` would cause all invocations of more , including invocations by programs such as man and msgs, to use this mode. Place the command sequence that sets up the MORE environment variable in your `.cshrc` or `.profile` file.

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

Other sequences that can be typed when more pauses, and their effects, are as follows (*i* is an optional integer argument, defaulting to one):

i<space> displays *i* more lines, (or another screenfull if no argument is given).

*i***b** skip back *i* screenfulls and display a screenfull of lines.

*i*CTRL B same as *i***b**.

*i*CTRL D displays 11 more lines (a "scroll"). If *i* is given, the scroll size is set to *i*.

*i***f** skips *i* screenfulls and displays a screenfull of lines.

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

*i***n** searches for the *i*-th occurrence of the last regular expression entered.

*i***s** skips *i* lines and displays a screenfull of lines.

*i***z** same as typing a space except that *i*, if present, becomes the new window size.

i:**n** skips to the *i*-th next file given in the command line (skips to last file if n doesn't make sense)

i:**p** skips to the *i*-th previous file given in the command line. If this command is given in the middle of displaying a file, more goes back to the beginning of the file. If *i* doesn't make sense, more skips back to the first file. If more is not reading from a file, the bell is rung and nothing else happens.

d same as CTRL D

h gives a description of all the more commands.

q Q exits from more.

v starts up the editor *vi* at the current line.

:f displays the current file name and line number.

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

= displays the current line number.

' (single quote) goes to the point from which the last search started. If no search has been performed in the current file, go back to the beginning of the file.

!command invokes a shell with *command*. The characters percent sign (%) and exclamation point (!) in *command* are replaced with the current file name and the previous shell command respectively. If there is no current file name, per cent sign is not expanded. The sequences \% and \! are replaced by % and ! respectively.

. (period) repeat the previous command.

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

At any time when output is being sent to the terminal, you can press the quit key (normally CTRL\). More stops sending output, and displays the usual --More-- prompt. The user can then enter one of the above commands in the normal manner. Unfortunately, some output is lost when this is done, due to the fact that any characters waiting in the terminal's output queue are flushed when the quit signal occurs.

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

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

A sample usage of more in previewing nroff output would be

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

NOTES

- Backwards scrolling of large files can be quite slow.

FILES

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

SEE ALSO

page, csh, msgs, script, sh

msgs

read system messages

The msgs command is used to read system messages (short pieces of information that are suitable to be read once by most users of the system.). Msgs is normally invoked by placing the command msgs -f in your .login or .profile file. Msgs prompts you with the source and subject of each new message. (If there is no subject line, the first few non-blank lines of the message are displayed.) If there is more to the message, you are told how long it is and asked whether you wish to see the rest of the message. The responses are listed below.

---

msgs [-fhlopq] [*number*] [*-number*]

msgs -s

msgs -c [*-days*]

ARGUMENTS

-f suppresses the "No new messages" display which otherwise occurs when there are no new messages. (useful in your .login file; see also -q).

-h displays the first part of messages only.

-l causes only locally originated messages to be reported.

-o causes *msgs* to prompt, rather than exit, when you read the last message.

-p causes long messages to be piped through more.

-q queries whether there are messages, displaying There are new messages. The command msgs -q is often used in login scripts.

number starts at the specified message rather than at the message indicated by your .msgsrc file. Thus, msgs -h 1 displays the first part of all messages.

-number starts the specified *number* of messages back from the one indicated by your .msgsrc file, useful for reviews of recent messages.

-s adds a new message to the messages directory (superuser only).

-c [*-days*] cleans up the messages directory, going back the specified number of *days* (superuser only). If not specified, *days* defaults to 21.

RESPONSES

The valid responses to the msgs prompt are:

y	types the rest of the message
RETURN	is equivalent to y.
p	types the rest of the message and enables "pipe through more" mode.
P	types the rest of the message and disables "pipe through more" mode.
n	skips this message and go on to the next message.
-	redisplays the last message.
n	goes to message number *n*.
q	exits from *msgs* skipping the next message.
x	exits from msgs without skipping the next message (the next time you run the program it picks up where you left off).
s	appends the current message to the file Messages in the current directory; s- saves the previously displayed message. A s or s- can be followed by a space and a filename replacing the default Messages.
m	or m- causes a copy of the specified message to be placed in a temporary mailbox and mail to be invoked on that mailbox. Both m and s accept a numeric argument in place of the hyphen.

DESCRIPTION

Msgs keeps track of the messages you have read by storing a number in the file .msgsrc in your home directory. The messages themselves are stored in individual files in the directory /usr/msgs. The file names are the (sequential) numbers of the messages they represent. The file /usr/msgs/bounds shows the low and high number of the messages in the directory so that msgs can quickly determine if there are messages that you have not read: If the contents of bounds is incorrect it can be fixed by removing it; msgs makes a new bounds file the next time it is run.

To add a new message to the directory you must have superuser privileges. Use the command msgs -s or mail the message to msgs (assuming that the following alias is established in /usr/lib/aliases):

```
msgs: "|/usr/ucb/msgs -s"
```

FILES

/usr/msgs/*	database
~/.msgsrc	number of next message to be presented

SEE ALSO

mail, more

mt

magnetic tape manipulating program

The mt command gives commands to a magnetic tape drive.

---

mt [**-f** *tapename*] *command* [*count*]

ARGUMENTS

-f *tapename*	specifies a magnetic tape drive (must reference a raw (not block) tape device). By default, the environment variable TAPE is used; if TAPE does not exist, mt uses the device /dev/rmt12.
command	is one of the commands listed below.
count	performs operations multiple times. By default *mt* performs the requested operation once.

DESCRIPTION

Only as many characters as are required to uniquely identify a command need be specified.

bsf	moves back space *count* files.
bsr	moves back space *count* records.
eof	writes *count* end-of-file marks at the current position on the tape.
erase	erases the entire cartridge. (Effective only on VMEbus SCSI devices.) This process takes over two hours on an Exabyte cartridge. See *Notes*.
fsf	moves forward space *count* files.
fsr	moves forward space *count* records.
offline	rewinds the tape and place the tape unit off-line (*count* is ignored.)
reten	retensions the tape in the cartridge. (Effective only on VMEbus SCSI devices.)
rewind	rewinds the tape (*count* is ignored.)
rewoffl	rewinds the tape and place the tape unit off-line (*count* is ignored.)
status	displays status information about the tape unit.
vstatus	displays status information about the Exabyte drive.

weof writes *count* end-of-file marks at the current position on the tape.

Mt returns a 0 exit status when the operation(s) were successful, 1 if the command was unrecognized, and 2 if an operation failed.

NOTES

- The command `mt -f /dev/rschs erase` (erase an Exabyte cartridge) takes up to two and one-half hours to complete. As it is an atomic driver command, it can not be aborted. It ignores CTRL C and CTRL Z. The only way to stop it is to reboot the system. It can be done in the background, so that it will not jam the terminal, but it still can not be killed.

FILES

`/dev/rmt*`	Raw magnetic tape interface
/dev/rschs	Exabyte cartridge tape drive

SEE ALSO

dd
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: mtio
Concentrix-2800 System Reference Manual: System Calls and C Library Routines: ioctl

mv

move or rename files

The mv command moves a file (plain file or directory) by changing its name or the directory containing it. Mv refuses to move a file onto itself.

---

mv [-i] [-f] [-] *file1 file2*

mv [-i] [-f] [-] *file1... directory*

ARGUMENTS

-i (interactive) prompts with a file name followed by a question mark whenever a move is to supercede an existing file. To continue, type y. Any other reply prevents the move.

-f (force) overrides any mode restrictions or the -i option.

- interprets all the following arguments to mv as file names. This allows file names starting with a hyphen.

file1 moves to *file2* or *directory*.

file2 removes *file2*, if it already exists, before *file1* is moved. If *file2* has a mode that forbids writing, mv displays the mode (see chmod *(System Calls)*) and reads the standard input to obtain a line; if the line begins with y, the move takes place; if not, mv exits.

directory moves *file1* to the specified *directory* with its original filename.

NOTES

- If *file1* and *file2* lie on different file systems, mv must copy the file and delete the original. In this case the owner name becomes that of the copying process and any linking relationship with other files is lost.

SEE ALSO

cp, ln

netstat

show network status

The netstat command symbolically displays various network-related information.

---

netstat [**-Aan**] [**-f** *address_family*] [*system*] [*core*]

netstat [**-himnrs**] [**-f** *address_family*] [*system*] [*core*]

netstat [**-n**] [**-I** *interface*] *interval* [*system*] [*core*]

ARGUMENTS

-A shows the address of any associated protocol control blocks; used for debugging.

-a shows the state of all sockets; normally sockets used by server processes are not shown.

-h shows the state of the IMP host table.

-I shows information only about this *interface*. Used with an *interval*, described below.

-i shows the state of interfaces that have been autoconfigured (interfaces statically configured into a system, but not located at boot time are not shown).

-m shows statistics recorded by the memory management routines (the network manages a private pool of memory buffers).

-n shows network addresses as numbers (normally netstat interprets addresses and attempts to display them symbolically). This option may be used with any of the display formats.

-r shows the routing tables. When **-s** is also present, show routing statistics instead.

-s shows per-protocol statistics.

-f *address_family*
Limit statistics or address control block reports to those of the specified address family. The following address families are recognized: inet, for AFINET, ns, for AFNS, and unix, for AFUNIX.

interval causes netstat to continuously display the information regarding packet traffic on the configured network interfaces, pausing *interval* seconds before refreshing the screen.

system specifies a substitute for the default /vmunix.

core specifies a substitute for the default /dev/kmem.

DESCRIPTION

The netstat command symbolically displays the contents of various network-related data structures. There are a number of output formats, depending on the options for the information presented. The first form of the command displays a list of active sockets for each protocol. The second form presents the contents of one of the other network data structures according to the option selected. Using the third form, with an interval specified, netstat will continuously display the information regarding packet traffic on the configured network interfaces.

The default display, for active sockets, shows the local and remote addresses, send and receive queue sizes (in bytes), protocol, and the internal state of the protocol. Address formats are of the form host.port or network.port if a socket's address specifies a network but no specific host address. When known the host and network addresses are displayed symbolically according to the data bases /etc/hosts and /etc/networks, respectively. If a symbolic name for an address is unknown, or if the -n option is specified, the address is printed numerically, according to the address family. For more information regarding the Internet dot format, refer to inet(C Routines). Unspecified, or wildcard, addresses and ports appear as "*".

The interface display provides a table of cumulative statistics regarding packets transferred, errors, and collisions. The network addresses of the interface and the maximum transmission unit (mtu) are also displayed.

The routing table display indicates the available routes and their status. Each route consists of a destination host or network and a gateway to use in forwarding packets. The flags field shows the state of the route (U if up), whether the route is to a gateway (G), and whether the route was created dynamically by a redirect (D). Direct routes are created for each interface attached to the local host; the gateway field for such entries shows the address of the outgoing interface. The *refcnt* field gives the current number of active uses of the route. Connection oriented protocols normally hold on to a single route for the duration of a connection while connectionless protocols obtain a route while sending to the same destination. The use field provides a count of the number of packets sent using that route. The interface entry indicates the network interface utilized for the route.

When netstat is invoked with an interval argument, it displays a running count of statistics related to network interfaces. This display consists of a column for the primary interface (the first interface found during autoconfiguration) and a column summarizing information for all interfaces. The primary interface may be replaced with another interface with the -I option. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.

EXAMPLE

The following example shows the output on an FX/2800 system for the command netstat –m (note the list of the cluster mbufs used/allocated):

```
% netstat -m
1085/1536 mbufs in use:
        324 mbufs allocated to data
        82 mbufs allocated to packet headers
        283 mbufs allocated to socket structures
        386 mbufs allocated to protocol control blocks
        6 mbufs allocated to routing table entries
        2 mbufs allocated to socket names and addresses
        2 mbufs allocated to interface addresses
36/192 half pages in use (low is 0)
384 Kbytes of mbufs allocated to network (70% in use)
384 Kbytes of half-pages allocated to network (18% in use)
768 Kbytes allocated to network (44% in use)
0 requests for memory denied
0 requests for memory delayed
0 calls to protocol drain routines
```

NOTES

- On systems running Network Information Service, the performance of the netstat command improves dramatically by using the **–n** option. Without this option, the system makes several NIS queries per socket entry in order to display the names of the hosts, networks, and ports, rather than the numeric values read from the local system. As a result, the performance of netstat on an NIS client is most closely related to network load.
- The notion of errors is ill-defined.
- Collisions mean something else for the IMP.

SEE ALSO

vmstat
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: hosts, networks, protocols, services
Concentrix-2800 System Administration: trpt

neqn

mathematics preprocessor for nroff (unsupported)

The neqn command invokes an nroff preprocessor for formatting mathematics on terminals, line printers, and so forth.

---

neqn [**-d***xy*] [**-p***n*] [**-s***n*] [**-f***n*] [*file...*]

ARGUMENTS

-d*xy* specifies that the characters *x* and *y* are equation delimiters. The left and right delimiters can be identical.

-p*n* controls the point size of subscripts and superscripts (reduced by three point sizes from the previous size by default).

-s*n* controls type size.

-f*n* controls the font.

file if no files are specified, the program reads from the standard input.

NOTES

- To embolden digits, parens, etc., it is necessary to quote them, as in `bold "12.3"`.

SEE ALSO

eqn, checkeq, nroff, tbl
Document Preparation Guide: EQN – Mathematics Typesetting
Document Preparation Guide: Using NROFF/TROFF

nice

set priority for command (Bourne shell)

The nice command executes a command with a specific scheduling priority.

---

nice [*-number*] *command*

ARGUMENTS

-number specifies a priority increment up to a limit of 20 (default 10). The *larger* the *number*, the *less* access a process has to system resources. The super-user can specify a negative *number* in order to increase access to resources. For example:

```
nice --10 ps alx
```

command specifies a command (and arguments) to execute.

DIAGNOSTICS

- Nice returns the exit status of the subject command.

SEE ALSO

nohup
Concentrix-2800 System Reference Manual: System Calls and C Library Routines: setpriority
Concentrix-2800 System Administration: renice

nice

set priority for command (C shell)

The nice command executes a command with a specific scheduling priority.

---

nice [*+number*] [*command*]

nice *–number* [*command*]

ARGUMENTS

+number specifies a priority increment up to a limit of 20 (default +4). The *larger* the *number*, the *less* access a process has to system resources.

–number specifies a priority decrement. The *smaller* the *number*, the *more* access a process has to system resources. Only the super-user can specify a negative *number*.

command specifies a command (and arguments) to execute. If you omit *command*, nice defaults to the current shell.

DIAGNOSTICS

- Nice returns the exit status of the subject command.

SEE ALSO

nohup
Concentrix-2800 System Reference Manual: System Calls and C Library Routines: setpriority
Concentrix-2800 System Administration: renice

nm

dump object file symbol table

The nm command displays the name list (symbol table) of an object file.

---

nm [-agnopru] [*file...*]

ARGUMENTS

-a prints symbol table entries inserted for use by debugger.

-g displays only global (external) symbols.

-n sorts numerically rather than alphabetically.

-o prepends file or archive name to each output line rather than only once (useful when processing output from a set of files via *grep*).

-p prevents sorting; displays in symbol-table order.

-r sorts in reverse order.

-u displays only undefined symbols.

file specifies an object file or an archive. If an archive, *nm* produces a listing for each object file in the archive. The default is `a.out`.

DESCRIPTION

Each symbol name is preceded by its value (blanks if undefined) and one of the letters;

U	(undefined)
A	(absolute)
T	(text segment symbol)
D	(data segment symbol)
B	(bss segment symbol)
C	(common symbol)
f	file name
-	debugger symbol table entry

If the symbol is local (non-external) the type letter appears in lower case. The output is sorted alphabetically.

SEE ALSO

ar
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: ar, a.out, stab

nohup

ignore hangup signals (Bourne shell)

The nohup command executes a specified command and makes it immune to hangup and terminate signals from the controlling terminal. The priority is incremented by five (resulting in reduced access to system resources). The standard output and standard error output are redirected to nohup.out.

/usr/bin/nohup *command*

ARGUMENTS

command specifies a command (and arguments) to execute.

NOTES

- Invoke nohup from the shell with ampersand (&) in order to prevent it from responding to interrupts by or stealing the input from the next person who logs in on the same terminal.
- The Bourne shell nohup command is not a builtin command; the C Shell version is.

FILES

nohup.out	standard output and standard error file under nohup.

SEE ALSO

nice, nohup (1c)
Concentrix-2800 System Reference Manual: System Calls and C Library Routiness: setpriority
Concentrix-2800 System Administration: renice

nohup

ignore hangup signals (C shell)

The nohup command executes a specified command and makes it immune to hangup signals from the controlling terminal. All background commands are immune to hangup signals by default.

---

nohup [*command*]

ARGUMENTS

command specifies a command (and arguments) to execute. In a shell script, you can omit *command* in order to make the remainder of the script immune to hangups.

NOTES

- The Bourne shell /usr/bin/nohup command is not a builtin command; the C Shell version is.

SEE ALSO

nice, nohup
Concentrix-2800 System Reference Manual: System Call and C Library Routiness: setpriority
Concentrix-2800 System Administration: renice

notify

display job status changes (C shell)

The notify command informs you asynchronously when the status of the current job (or a specified job) changes. Normally, the notification appears before a shell prompt.

---

notify [*%job*]...

ARGUMENTS

%job specifies a job number.

NOTES

- Notification is automatic if the shell variable `notify` is set.

SEE ALSO

jobs

nroff

text formatter (unsupported)

The nroff command formats the text in the named files for terminals, line printers, and similar devices.

---

nroff [*option...*] [*file...*]

ARGUMENTS

The options can appear in any order so long as they appear *before* the files:

-o*list* prints only pages whose page numbers appear in the comma-separated *list* of numbers and ranges. A range *N–M* means pages *N* through *M*; an initial *–N* means from the beginning to page *N*; and a final *N–* means from *N* to the end.

-n*N* numbers the first page *N*.

-s*N* stops every *N* pages (default *N* is every page) to allow paper loading or changing, and resumes upon receipt of a newline.

-m*name* prepends the macro file `/usr/lib/tmac/tmac.`*name* to the input *files*.

-r*aN* sets register *a* (one-character) to *N*.

-i reads from the standard input after the input files are exhausted.

-q Invoke the simultaneous input-output mode of the *rd* request.

-T*name* prepares output for specified terminal. Known *names* are:

37	(default) Teletype Corporation Model 37 terminal.
tn300	GE TermiNet 300 (or any terminal without half-line capability).
300S	DASI-300S.
300	DASI-300.
450	DASI-450 (Diablo Hyterm).

-e produces equally-spaced words in adjusted lines, using full terminal resolution.

-h uses output tabs during horizontal spacing to speed output and reduce output character count. Tab settings are assumed to be every eight nominal character widths.

file specifies an input file. If not present, the standard input is read. A single hyphen specifies the standard input.

FILES

/tmp/ta*	temporary file
/usr/lib/tmac/tmac.*	standard macro files
/usr/lib/term/*	terminal driving tables for nroff

SEE ALSO

troff, eqn, neqn, tbl, col, deroff
Document Preparation Guide: Using NROFF/TROFF, TROFF Tutorial

od

octal, decimal, hex, ASCII dump

The od command displays a file in one or more dump formats.

---

od [*-format*] [*file*] [[+] *offset* [.] [b] [*label*]]

ARGUMENTS

-format specifies a dump format.

a interprets bytes as characters and displays them with their ASCII names. The parity bit is ignored.

ap interprets bytes as characters and displays them with their ASCII names. Bytes with even parity are underlined.

aP interprets bytes as characters and displays them with their ASCII names. Bytes with odd parity to be underlined.

b interprets bytes as unsigned octal.

B interprets words as unsigned octal.

c interprets bytes as ASCII characters. Certain nongraphic characters appear as C escapes: null=\0, backspace=\b, formfeed=\f, newline=\n, return=\r, tab=\t; others appear as three-digit octal numbers. Bytes with the parity bit set are displayed in octal.

d interprets (short) words as unsigned decimal.

D interprets longwords as unsigned decimal.

f interprets longwords as single precision floating point.

F interprets quadwords as double precision floating point.

h interprets (short) words as unsigned hexadecimal.

H interprets longwords as unsigned hexadecimal.

i interprets (short) words as signed decimal.

I interprets longwords as signed decimal.

l interprets longwords as signed decimal.

L interprets longwords as signed decimal.

o interprets (short) words as unsigned octal. This is the default format.

O interprets (long) words as unsigned octal.

s[n] looks for strings of ASCII graphic characters, terminated with a null byte. *N* specifies the minimum length string to be recognized. By default, the minimum length is three characters.

v shows all data. By default, display lines that are identical to the last line shown are not output, but are indicated with an asterisk in column one.

V prints an extra space between labels, improving readability.

w[n] specifies the number of input bytes to be interpreted and displayed on each output line. If w is not specified, 16 bytes are read for each display line. If *n* is not specified, it defaults to 32.

x interprets (short) words as hexadecimal.

X interprets longwords as hexadecimal.

file specifies a file to be displayed. The default is the standard input.

+ if *file* is omitted, an *offset* argument must be preceded by a plus sign.

offset specifies the byte offset into the file where dumping is to commence. By default this argument is interpreted in octal. A different radix can be specified:

. if a period is appended to the argument, *offset* is interpreted in decimal.

x 0x if *offset* begins with x or 0x, it is interpreted in hexadecimal.

b if b(B) is appended, the offset is interpreted as a block count, where a block is 512 (1024) bytes.

The radix of the displayed address is the same as the radix of the *offset*, if specified; otherwise it is octal.

label specifies a pseudoaddress for the first byte displayed. It is shown in parentheses following the file offset and is intended to be used with core images to indicate the real memory address. The syntax for *label* is identical to that for *offset*.

FX/2800 VERSUS FX/80 DISPLAYS

The od command displays files in various formats (octal, hexadecimal, decimal, floating point, and so on) and various increments (byte, word, longword, and so on). Starting at the beginning of the file (which corresponds to the low-order position of memory storage), entities are displayed from left to right, and top to bottom. However, each entity is

displayed so that the most significant digit is leftmost, in keeping with standard mathematical practice. Because the FX/2800 Series machine is big endien, octal and hexadecimal displays appear reversed from FX/80 Series displays as the increment increases, which may confuse the viewer.

For example, if a file contains 2 longword integers with values of 1 and 2, the od displays in longword formats such as -d (longword decimal) and -X (longword hexadecimal) are the same on both machines. The display consists of 2 entities: the leftmost entity represents the first longword in the file; the next entity represents the second longword in the file. (The column to the left contains the address of the first entity on the line in octal.)

```
% od -d testfile
0000000           1           2
0000010

% od -X testfile
0000000  00000001 00000002
0000010
```

If the file is displayed in -b (byte, octal) format, each byte appears in its order of storage. On the FX/80 Series machine, the storage of each longword is from most signficant byte to least significant byte, so the display appears as follows:

```
% od -b testfile                                    (FX/80)
0000000  000 000 000 001 000 000 000 002
0000010
```

On the FX/2800 Series machine, the storage of each longword is from least significant byte to most significant byte:

```
% od -b testfile                                    (FX/2800)
0000000  001 000 000 000 002 000 000 000
0000010
```

Likewise, in -x (word, hexadecimal) format, the storage of each longword starts with the most significant word on the FX/80 Series machine, and the least significant word on the FX/2800 Series machine:

```
% od -x testfile                                    (FX/80)
0000000  0000 0001 0000 0002
0000010

% od -x testfile                                    (FX/2800)
0000000  0001 0000 0002 0000
0000010
```

NOTES

- A file name argument cannot start with a plus sign.
- A hexadecimal offset cannot be a block count.
- Only one file name argument can be given.

SEE ALSO

adb

on

execute a command remotely

The on command executes commands on another system.

---

on [**-i**] [**-n**] [**-d**] *host command* [*argument*] ...

ARGUMENTS

-i Interactive mode: use remote echoing and special character processing. This option is needed for programs that expect to be talking to a terminal. All terminal modes and window size changes are propagated.

-n No Input: this option causes the remote program to get end-of-file when it reads from standard input, instead of passing standard input from the standard input of the *on* program. For example, -n is necessary when running commands in the background with job control.

-d Debug mode: print out some messages as work is being done.

DESCRIPTION

The on program is used to execute commands on another system, in an environment similar to that invoking the program. All environment variables are passed, and the current working directory is preserved. To preserve the working directory, the working file system must be either already mounted on the host or be exported to it. Relative path names will only work if they are within the current file system; absolute path names may cause problems.

Standard input is connected to standard input of the remote command, and standard output and standard error from the remote command are sent to the corresponding files for the *on* command.

SEE ALSO

Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: exports,
Concentrix-2800 System Administration: rexd

DIAGNOSTICS

- unknown host: Host name not found
- cannot connect to server: Host down or not running
- the server can't find .: Problem finding the working directory
- can't locate mount point: Problem finding current file system

Other error messages may be passed back from the server.

onintr

enable interrupt handling (C shell)

The onintr command enables interrupt handling. It is intended for use in shell scripts. For example, a script that creates temporary files might want to handle interrupts in order to clean up after itself.

---

onintr [**-**]
onintr *label*

ARGUMENTS

With no arguments, onintr restores the default action of the shell on interrupts, which is to terminate shell scripts or to return to the terminal command input level.

- causes all interrupts to be ignored.

label when an interrupt is received or a child process terminates because it was interrupted, the shell transfers control to the command following the specified *label*. Interrupt handling remains in effect; thus, the subsequent commands (the interrupt handler) are responsible for exiting from the shell script.

EXAMPLE

This shell script contains an infinite loop and an interrupt handler:

```
# idemo
onintr ihand
while (1)
        echo foobar
end
ihand:
echo signal received
exit
```

Running the script produces:

```
% idemo
foobar
foobar
foobar
^Csignal received
%
```

NOTES

- If the shell is running detached and interrupts are being ignored, onintr has no meaning and interrupts continue to be ignored by the shell and all invoked commands.

SEE ALSO

exit
System User's Guide: Bourne Shell Reference: trap

page

file perusal filter for crt viewing

The page command displays a text file one screenful at a time on a video terminal. It normally pauses after each screenful, displaying --More-- at the bottom of the screen. Pressing RETURN displays one more line. Pressing the space bar clears the screen and displays another screenful (but only if a full screenful is being displayed), and $k - 1$ lines are displayed, where k is the number of lines the terminal can display.

---

page [-cdflsu] [-*windowsize*] [+*linenumber*] [+/*pattern*] [*name...*]

ARGUMENTS

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

-d prompts with the message `Hit space to continue, Rubout to abort` at the end of each screenful.

-f counts logical, rather than screen lines (long lines are not folded). This option is recommended if nroff output is being piped through ul, which can generate escape sequences containing characters that would ordinarily occupy screen positions, but that do not display as part of an escape sequence. That can cause *page* to fold lines erroneously because they appear to be longer than they actually are.

-l suppresses special treatment of formfeeds (CTRL L). By default, page pauses after any line that contains a formfeed, as if the end of a screenful had been reached. If a file begins with a formfeed, the screen is cleared before the file is displayed.

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

-u suppresses processing of underlining. By default, if the terminal can perform underlining or has a stand-out mode, page outputs appropriate escape sequences for underlined information in the source file.

-windowsize
is an integer that specifies the size (in lines) of the window that *page* uses.

+linenumber
specifies the line number at which to start up.

+/pattern
causes page to start up two lines before the line containing the specified regular expression.

DESCRIPTION

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

Page looks in the environment variable MORE to preset any options desired. For example, if you prefer to view files using the *-c* mode of operation, the csh command setenv MORE -c or the sh command sequence MORE='-c' ; export MORE would cause all invocations of page , including invocations by programs such as man and msgs, to use this mode. Place the command sequence that sets up the MORE environment variable in your .cshrc or .profile file.

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

Other sequences that can be typed when page pauses, and their effects, are as follows (*i* is an optional integer argument, defaulting to 1) :

i<space>
displays *i* more lines, (or another screenful if no argument is given).

CTRL D displays 11 more lines (a "scroll"). If *i* is given, the scroll size is set to *i*.

d same as CTRL D

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

is skips *i* lines and displays a screenful of lines.

if skips *i* screenfuls and displays a screenful of lines.

q Q exits from *page*.

= displays the current line number.

v starts up the editor *vi* at the current line.

h gives a description of all the page commands.

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

in searches for the *i*-th occurrence of the last regular expression entered.

' (single quote) goes to the point from which the last search started. If no search has been performed in the current file, go back to the beginning of the file.

!*command*
invokes a shell with *command*. The characters percent sign (%) and exclamation point (!) in *command* are replaced with the current file name and the

previous shell command respectively. If there is no current file name, per cent sign is not expanded. The sequences \% and \! are replaced by % and ! respectively.

i:n skips to the *i*–th next file given in the command line (skips to last file if n doesn't make sense)

i:p skips to the *i*–th previous file given in the command line. If this command is given in the middle of displaying a file, *page* goes back to the beginning of the file. If *i* doesn't make sense, page skips back to the first file. If page is not reading from a file, the bell is rung and nothing else happens.

:f displays the current file name and line number.

:q :Q exits from *page* (same as q or Q).

. (period) repeat the previous command.

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

At any time when output is being sent to the terminal, you can press the quit key (normally CTRL \). Page stops sending output, and displays the usual --More-- prompt. The user can then enter one of the above commands in the normal manner. Unfortunately, some output is lost when this is done, due to the fact that any characters waiting in the terminal's output queue are flushed when the quit signal occurs.

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

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

A sample usage of page in previewing nroff output follows:

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

FILES

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

SEE ALSO

more, csh, msgs, script, sh

pagesize

display system page size

The pagesize command displays the size of a page of memory in bytes, as returned by the system call getpagesize. This program is useful in constructing portable shell scripts.

---

pagesize

SEE ALSO

Concentrix-2800 System Reference: System Calls and C Library Routines: getpagesize

passwd

change login password, finger information, or default shell

The passwd command changes (or installs) user information. *Passwd* now includes all the functions that were performed separately by the passwd, chfn and chsh commands.

---

passwd [-f] [-s] [*name*]

ARGUMENTS

-f changes the GECOS (finger) information.

-s changes the default shell.

name specifies a user name (your own name by default).

DESCRIPTION

This command changes (or installs) a password, GECOS information field (-f option), or login shell (-s option) associated with the user name. If you don't enter a user name, your own name is used by default.

When altering a password, the program prompts for the current password and then for the new one. The caller must supply both. The new password must be typed twice to forestall mistakes. New passwords must be at least four characters long if they use a sufficiently rich alphabet and at least six characters long if monocase. These rules are relaxed if you are insistent enough. Only the owner of the name or the super-user may change a password; the owner must prove he knows the old password.

When altering the GECOS information field, passwd displays the current information, broken into fields, as interpreted by the finger program, among others, and prompts for new values. These fields include a user's "real life" name, office room number, office phone number, and home phone number. Included in each prompt is a default value, which is enclosed between brackets. The default value is accepted simply by typing a carriage return. To enter a blank field, the word "none" may be typed. Below is a sample run:

```
Name [Biff Studsworth II]:
Room number (Exs: 597E or 197C) []: 521E
Office Phone (Ex: 1632) []: 1863
Home Phone (Ex: 987532) [5771546]: none
```

Passwd allows phone numbers to be entered with or without hyphens. It is a good idea to run finger after changing the GECOS information to make sure everything is setup properly.

The superuser may change anyone's GECOS information; normal users may only change their own.

When altering a login shell, passwd displays the current login shell and then prompts for the new one. The new login shell must be one of the approved shells listed in/etc/shells unless you are the superuser. If /etc/shells does not exist, the only shells that may be specified are /bin/sh and /bin/csh.

The superuser may change anyone's login shell; normal users may only change their own login shell.

NOTES

- Use yppasswd to change your password in the NIS. This will not affect your local password, or your password on any remote machines on which you have accounts.
- The commands passwd -f and passwd -s will only affect the entries in the local passwd file, and will not affect the entries in any NIS passwd file.

FILES

/etc/passwd	the file containing all of this information
/etc/shells	the approved list of shells

SEE ALSO

login, finger, yppasswd
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: passwd
Concentrix-2800 System Reference: System Calls and C Library Routines: crypt
UNIX Password Security, Robert Morris and Ken Thompson

percent_sign

place job in foreground/background (C shell)

The per cent sign (%) introduces a job name and is a synonym for fg (or bg if an ampersand is supplied).

%job [&]

ARGUMENTS

job specifies a job; it can be one of the following:

n where *n* is the job number as shown by the jobs command.

string where *string* is an unambiguous prefix of the string typed to start the job. For example, %ex would normally restart a suspended ex job, if there were only one suspended job whose name began with the string ex.

?*string* where *string* is an unambiguous substring of the string typed to start the job. For example, %ex would normally restart a suspended *ex* job, if there were only one suspended job whose name began with the string ex.

+ specifies the current job as shown by the jobs command.

% specifies the current job as shown by the jobs command (for close analogy with the syntax of the history mechanism).

- specifies the previous job as shown by the jobs command.

& resumes the job in the background; if omitted, the job is resumed in the foreground.

EXAMPLE

`%1` brings job 1 into the foreground (same as `fg %1`).

SEE ALSO

bg, fg, jobs
Concentrix System User's Guide: Using the C Shell

plot

graphics filters

The plot command reads plotting instructions from the standard input, or the specified files, and produces plotting instructions suitable for a particular terminal on the standard output.

---

plot [-T*term*] [-r*resolution*] [*files...*]

ARGUMENTS

-T *term* specifies a terminal type. If not specified, the environment parameter $TERM is used by default. Known *terminals* are:

4013 Tektronix 4013 storage scope.

4014 or **tek**

Tektronix 4014 or 4015 storage scope with Enhanced Graphics Module. Use **4013** for a 4014 or 4015 without Enhanced Graphics Module.

450 DASI Hyterm 450 terminal (Diablo mechanism).

300 DASI 300 or GSI terminal (Diablo mechanism).

300S DASI 300S terminal (Diablo mechanism).

aed AED 512 color graphics terminal.

bitgraph or **bg**

BBN bitgraph graphics terminal.

imagen or **ip**

Imagen laser printer (default 240 dpi resolution).

crt Any crt capable of running *vi*.

dumb Dumb terminals without cursor addressing, also line printers.

vt125 DEC VT125 terminals.

hp2648 or **hp** or **hp8**

Hewlett Packard 2648 graphics terminal.

var Benson Varian printer-plotter. Sends the result directly to the plotter device, rather than to standard output.

ver Versatec D1200A printer-plotter. Sends the result directly to the plotter device, rather than to standard output.

-rresolution specifies the device's output resolution. So far, only the Imagen laser printer understands this option.

files... specifies files for *plot* to take input from.

DESCRIPTION

A graphics file is a stream of plotting instructions. Each instruction consists of an ASCII letter usually followed by bytes of binary information. The instructions are executed in order. A point is designated by four bytes representing the x and y values; each value is a signed integer. The last designated point in an l, m, n, or p instruction becomes the 'current point' for the next instruction.

Each of the following descriptions begins with the name of the corresponding routine in the *plot* subroutine package.

m move: The next four bytes give a new current point.

n cont: Draw a line from the current point to the point given by the next four bytes.

p point: Plot the point given by the next four bytes.

l line: Draw a line from the point given by the next four bytes to the point given by the following four bytes.

t label: Place the following ASCII string so that its first character falls on the current point. The string is terminated by a newline.

a arc: The first four bytes give the center, the next four give the starting point, and the last four give the end point of a circular arc. The least significant coordinate of the end point is used only to determine the quadrant. The arc is drawn counter-clockwise.

c circle: The first four bytes give the center of the circle, the next two the radius.

e erase: Start another frame of output.

f linemod: Take the following string, up to a newline, as the style for drawing further lines. The styles are 'dotted,' 'solid,' 'longdashed,' 'shortdashed,' and 'dotdashed.' Effective only in *plot 4014* and *plot ver*.

s space: The next four bytes give the lower left corner of the plotting area; the following four give the upper right corner. The plot is magnified or reduced to fit the device as closely as possible.

Space settings that exactly fill the plotting area with unity scaling appear below for devices supported by the filters of *plot*. The upper limit is just outside the

plotting area. In every case the plotting area is taken to be square; points outside may be displayable on devices whose face isn't square.

```
4014            space(0, 0, 3120, 3120);
ver             space(0, 0, 2048, 2048);
300, 300s       space(0, 0, 4096, 4096);
450             space(0, 0, 4096, 4096);
```

NOTES

- There is no lockout protection for /usr/tmp/raster.

FILES

/usr/bin/tek
/usr/bin/t450
/usr/bin/t300
/usr/bin/t300s
/usr/bin/aedplot
/usr/bin/bgplot
/usr/bin/crtplot
/usr/bin/dumbplot
/usr/bin/gigiplot
/usr/bin/hpplot
/usr/bin/implot
/usr/ucb/lpr

SEE ALSO

graph, spline
Concentrix-2800 System Reference: System Calls and C Library Routines: plot
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: plot

popd

pop directory stack (C shell)

The popd command changes your current working directory to the most recent entry on the directory stack and pops the stack.

---

popd [*+n*]

ARGUMENTS

+n discards the *n*th entry in the stack. (The stack elements are numbered from zero, starting at the top.)

DIAGNOSTICS

- Popd displays the directory stack after completing its operation.

SEE ALSO

cd, dirs, pushd

pr

format and display file

The pr command produces a formatted listing of one or more files. The output is separated into pages headed by a date, the name of the file or a specified header, and the page number. Inter-terminal messages via write are disabled during a pr.

---

pr [*option...*] [*file...*]

ARGUMENTS

Options apply to all following files but can be reset between files:

-*n* produces *n*-column output.

+*n* begins at page *n*.

-h specifies that the subsequent argument is a page header.

-w*n* specifies the width of the page for multi-column output (*n* characters). The default is 72.

-f uses formfeeds instead of newlines to separate pages. A formfeed is assumed to use up two blank lines at the top of a page. (Thus, this option does not affect the effective page length.)

-l*n* specifies the length of the page (*n* lines). The default is 66.

-t disables the five-line header or the five-line trailer normally supplied for each page.

-sc separates columns by the single character *c* instead of by the appropriate amount of white space. The default character is a tab.

-m displays all *files* simultaneously, each in one column,

file specifies a file to display. If omitted, pr displays its standard input.

FILES

/dev/tty? to suspend messages.

SEE ALSO

cat

prelf

dump contents of an ELF file as text

The prelf command dumps the contents of an ELF file as readable text.

---

prelf [-n] [-w] [*elf-file*]

ARGUMENTS

-n forces 80 colum output.

-w forces 132 colum output.

elf-file specifies a file in ELF format. This may be an object file or an executable file.

DESCRIPTION

If neither the **-n** nor **-w** options is specified, the prelf command tries to use the window size of the user's terminal. If that size is unavailable, prelf uses 80.

EXAMPLES

These examples print an executable file on a printer:

```
% prelf -n /bin/ls | print
% prelf -w /bin/ls/prelf | print -Plpw
```

This example prints an executable file on standard output, piped through more:

```
% prelf /bin/ls | more
```

FILES

a.out

SEE ALSO

Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: a.out

print

pr to the line printer

The print command prints with pr, a copy of each named file on the line printer.

---

print *file...*

DESCRIPTION

Print is a one line shell script:

```
lpr -p $*
```

SEE ALSO

lpr, pr

printenv

display environment variables

The printenv command displays the values of the variables in the environment.

---

printenv [*name*]

ARGUMENTS

name specifies an environment variable. If not defined in the environment, printenv returns exit status 1, else it returns status 0.

SEE ALSO

setenv, unsetenv, sh, csh

prs

print an SCCS file

The prs command prints, on the standard output, parts or all of an SCCS file (see sccsfile *(System Reference Manual Vol. II)* in a user-supplied format.

---

prs [-d[*dataspec*]] [-r[*SID*]] [-e] [-l] [-a] file...

ARGUMENTS

Arguments to prs, which can appear in any order, consist of *keyletter* arguments, and file names. All the *keyletter* arguments apply independently to each named file:

-d[*dataspec*]
: specifies the output data specification. The *dataspec* is a string consisting of SCCS file *data keywords* (see *DATA KEYWORDS*) interspersed with optional user supplied text.

-r[*SID*]
: specifies the *S*CCS *ID*entification (SID) string of a delta for which information is desired. If no SID is specified, the SID of the most recently created delta is assumed.

-e
: requests information for all deltas created *earlier* than and including the delta designated via the -r keyletter.

-l
: requests information for all deltas created *later* than and including the delta designated via the -r keyletter.

-a
: requests printing of information for both removed, i.e., delta type = *R*, (see *rmdel*) and existing, i.e., delta type = *D*, deltas. If the -a keyletter is not specified, information for existing deltas only is provided.

file
: specifies a file to be processed. If a directory is named, *prs* behaves as though each file in the directory were specified as a named file, except that nonSCCS files (last component of the path name does not begin with s.), and unreadable files are silently ignored. If a name of "-" is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file or directory to be processed; non-SCCS files and unreadable files are silently ignored.

DATA KEYWORDS

Data keywords specify which parts of an SCCS file are to be retrieved and output. All parts of an SCCS file (see *System Reference Manual, Miscellaneous Routines, Special Files,and File Formats*: sccsfile) have an associated data keyword. There is no limit on the number of times a data keyword can appear in a *dataspec*.

The information printed by prs consists of: the user supplied text and appropriate values (extracted from the SCCS file) substituted for the recognized data keywords in the

order of appearance in the *dataspec*. The format of a data keyword value is either *Simple* (S), in which keyword substitution is direct, or *Multi-line* (M), in which keyword substitution is followed by a carriage return.

User supplied text is any text other than recognized data keywords. A tab is specified by \t and carriage return/new-line is specified by \n.

SCCS FILES DATA KEYWORDS

Keyword	Data Item	File Section	Value	Format
:Dt:	Delta information	Delta Table	See below*	S
:DL:	Delta line statistics	"	:Li:/:Ld:/:Lu:	S
:Li:	Lines inserted by Delta	"	nnnnn	S
:Ld:	Lines deleted by Delta	"	nnnnn	S
:Lu:	Lines unchanged by Delta	"	nnnnn	S
:DT:	Delta type	"	*D*-or-*R*	S
:I:	SCCS ID string (SID)	"	:R:.:L:.:B:.:S:	S
:R:	Release number	"	nnnn	S
:L:	Level number	"	nnnn	S
:B:	Branch number	"	nnnn	S
:S:	Sequence number	"	nnnn	S
:D:	Date Delta created	"	:Dy:/:Dm:/:Dd:	S
:Dy:	Year Delta created	"	nn	S
:Dm:	Month Delta created	"	nn	S
:Dd:	Day Delta created	"	nn	S
:T:	Time Delta created	"	:Th:::Tm:::Ts:	S
:Th:	Hour Delta created	"	nn	S
:Tm:	Minutes Delta created	"	nn	S
:Ts:	Seconds Delta created	"	nn	S
:P:	Programmer who created Delta	"	logname	S
:DS:	Delta sequence number	"	nnnn	S
:DP:	Predecessor Delta seq-no.	"	nnnn	S
:DI:	Seq-no. of deltas incl., excl., ignored	"	:Dn:/:Dx:/:Dg:	S
:Dn:	Deltas included (seq #)	"	:DS:-:DS:...	S
:Dx:	Deltas excluded (seq #)	"	:DS:-:DS:...	S
:Dg:	Deltas ignored (seq #)	"	:DS:-:DS:...	S
:MR:	MR numbers for delta	"	text	M
:C:	Comments for delta	"	text	M
:UN:	User names	User Names	text	M
:FL:	Flag list	Flags	text	M
:Y:	Module type flag	"	text	S
:MF:	MR validation flag	"	*yes*-or-*no*	S
:MP:	MR validation pgm name	"	text	S
:KF:	Keyword error/warning flag	"	*yes*-or-*no*	S
:BF:	Branch flag	"	*yes*-or-*no*	S
:J:	Joint edit flag	"	*yes*-or-*no*	S
:LK:	Locked releases	"	:R:...	S
:Q:	User defined keyword	"	text	S
:M:	Module name	"	text	S
:FB:	Floor boundary	"	:R:	S
:CB:	Ceiling boundary	"	:R:	S

:Ds:	Default SID	"	:I:	S
:ND:	Null delta flag	"	*yes*-or-*no*	S
:FD:	File descriptive text	Comments	text	M
:BD:	Body	Body	text	M
:GB:	Gotten body	"	text	M
:W:	A form of *what* string	N/A	:Z::M:\t:I:	S
:A:	A form of *what* string	N/A	:Z::Y:-:M:-:I::Z:	S
:Z:	*what* string delimiter	N/A	@(#)	S
:F:	SCCS file name	N/A	text	S
:PN:	SCCS file path name	N/A	text	S

```
* :Dt: = :DT: :I: :D: :T: :P: :DS: :DP:
```

EXAMPLES

The following command:

```
prs -d"Users and/or user IDs for :F: are:\n:UN:" s.file
```

produces on the standard output:

```
Users and/or user IDs for s.file are:
xyz
131
abc
```

The following command:

```
prs -d"Newest delta for pgm :M:: :I: Created :D: By :P:" -r s.file
```

produces on the standard output:

```
Newest delta for pgm main.c: 3.7 Created 77/12/1 By cas
```

As a special case, the following command:

```
prs s.file
```

produces the following on the standard output for each delta table entry of the "D" type. The only keyletter argument allowed to be used with the *special case* is the -a keyletter:

```
D 1.1 77/12/1 00:00:00 cas 1 000000/00000/00000
MRs:
bl78-12345
bl79-54321
COMMENTS:
this is the comment line for s.file initial delta
```

DIAGNOSTICS

- Use helpsccs for explanations.

FILES

/tmp/pr?????

SEE ALSO

sccs, admin, delta, get, help
Programming Tools Guide: Source Code Control System – System V Interface
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats sccsfile

ps

display process status information

The ps command displays information about processes.

---

ps [agltuwx#] [*image*]

ARGUMENTS

a displays information about all processes with terminals. By default, only one's own processes are displayed.

g displays all processes. By default, ps only displays "interesting" processes. Processes are deemed to be uninteresting if they are process group leaders. This normally eliminates top-level command interpreters and processes waiting for users to login on free terminals.

l displays a long listing, with fields UID, PPID, CP, C/P/NP PRI, NI, VSZ, RSS and SLEEP as described below.

t*x* restricts output to processes whose controlling tty is *x* (which should be specified as displayed by ps, that is, `t3` for tty3, `tco` for console, `td0` for ttyd0, `t?` for processes with no tty, `t` for processes at the current tty, etc). This option must appear last in the list.

u produces a user-oriented output; includes fields USER, VSZ and RSS as described below.

w overrides the terminal width and forces the output width limit to 132 columns, even on terminals which are narrower than this. This makes it possible to see the entire command name even when doing a ps –alx command, which usually truncates most commands.

x displays information about processes with no terminal.

restricts output to a specified process number. This option must appear last in the list.

image specifies the name of the system image to examine (default is /unix).

DESCRIPTION

All output formats include, for each process:

PID	process identification
TT	control terminal
STAT	current state
TIME	cpu time used (includes both user and system time)
COMMAND	an indication of the command that is running

The current state is shown as:

Run	runnable
Stop	stopped
Slp	sleeping
Idle	waiting to become runnable after process creation
Exit	waiting for parent process to read exit status prior to termination

The letter S following the state indicates that the process is currently swapped out.

Fields that are not common to all output formats:

USER — the name of the effective user-id of the process (see s protection under chmod).

NICE — (NI) the process scheduling increment (see setpriority:*Concentrix-2800 System Reference: System Calls and C Library Routines*).

VSZ — the virtual size of the process (in 1024-byte increments). The letter L appended to the virtual size means that the process references the shared library.

RSS — the real memory (resident set) size of the process (in 1024-byte increments)

UID — the numerical effective user–id of the process (see s protection under chmod)

PPID — the numerical id of the parent of the process.

CP — the short-term cpu utilization factor (used in scheduling).

C/P/NP — C is the scheduling class for the process. It is determined at execution time but can be changed by a privileged system call or command (see setclass). P is the priority of the process. The value of P ranges from 1–9. Zero is highest priority, nine is lowest. P increases as the process remains compute-bound. P decreases as the process becomes blocked on I/O. NP is the number of processors required by the process.

PRI — process priority (non-positive when in noninterruptible wait).

SLEEP — the event on which the process is waiting (an address in the system).

NOTES

- Things can change while ps is running; the picture it gives is only a close approximation to reality.
- With version 2.0 of Concentrix-2800, ps displays thread identifiers.

The following example demonstrates the execution of a program (named watch) that creates two threads. While the program is running in the background, the ps command displays the threads. The PID field for each thread displays the same process identifier. For the root thread, the COMMAND field displays the name of the command that started the process; for the other threads, the COMMAND field displays the thread identifier.

```
% watch&
func: this is thread 1
Thread id of thread 1: 28121
Thread id of thread 2: 3934
func: this is thread 2
[1] 32200
% ps

  PID TT STAT     TIME  COMMAND
19674 ra Run      0:00  ps
23904 ra Slp      0:00  -csh  (csh)
32200 ra Slp      0:31  watch
32200 ra Slp      0:00  [03934]
32200 ra Slp      0:00  [28121]
```

FILES

/vmunix	system namelist
/dev	searched to find tty names

SEE ALSO

kill, w,
Concentrix-2800 System Administration: thr_overview

ptx

permuted index (unsupported)

Ptx generates a permuted index to file input on file output (standard input and output default). It has three phases: the first does the permutation, generating one line for each keyword in an input line. The keyword is rotated to the front. The permuted file is then sorted. Finally, the sorted lines are rotated so the keyword comes at the middle of the page.

---

ptx [*option*] ... [*input* [*output*]]

ARGUMENTS

-b *break* Use the characters in the break file to separate words. In any case, tab, newline, and space characters are always used as break characters.

-f Fold upper and lower case letters for sorting.

-g *n* Use the next argument, *n*, as the number of characters to allow for each gap among the four parts of the line as finally printed. The default gap is 3 characters.

-i *ignore* Do not use as keywords any words given in the ignore file. If the -i and -o options are missing, use /usr/lib/eign as the ignore file.

-o *only* Use as keywords only the words given in the only file.

-r Take any leading nonblank characters of each input line to be a reference identifier (as to a page or chapter) separate from the text of the line. Attach that identifier as a 5th field on each output line.

-t Prepare the output for the phototypesetter; the default line length is 100 characters.

-w *n* Use the next argument, *n*, as the width of the output line. The default line length is 72 characters.

DESCRIPTION

Ptx produces output in the form:

```
.xx "tail" "before keyword" "keyword and after" "head"
```

where .xx may be an nroff or macro for user-defined formatting. The *beforekeyword* and *keywordand* fields incorporate as much of the line as will fit around the keyword when it is printed at the middle of the page. *Tail* and *head*, at least one of which is an empty string " ", are wrapped-around pieces small enough to fit in the unused space at the opposite end of the line. When original text must be discarded, '/' marks the spot.

FILES

/usr/bin/sort
/usr/lib/eign

NOTES

Line length counts do not account for overstriking or proportional spacing.

pushd

push directory onto stack (C shell)

The pushd command manipulates the directory stack.

---

pushd [*name*]

pushd +*n*

ARGUMENTS

With no arguments, *pushd* exhanges the two top elements of the directory stack.

name changes to *name* (the new directory) and pushes the old directory onto the stack.

+*n* rotates the *n*th element of the directory stack around to the top and changes to it. (The stack elements are numbered from zero, starting at the top.)

DIAGNOSTICS

- Pushd displays the directory stack after completing its operation.

SEE ALSO

dirs, popd

pwd

display working directory pathname

The pwd command displays the pathname of the working (current) directory.

---

pwd

NOTES

- In csh the command dirs is always faster (although it can give a different answer in the rare case that the current directory or a containing directory was moved after the shell descended into it).

SEE ALSO

cd, csh
Concentrix-2800 System Reference: System Calls and C Library Routines: getwd

qdel

delete batch request

The qdel program deletes batch system requests.

---

qdel [-k] *requestid...*

ARGUMENTS

-k kills (deletes) the specified *running* batch requests. Without this option, qdel kills only non-running requests.

requestid specifies a batch request in the form *username.number* as shown by qstat and as returned by the submitted qsub command.

DESCRIPTION

Qdel provides the ability to delete queued or running batch requests. In order to change the nature of a batch request, for example, the procedure is to first delete the request and then resubmit it with the desired attributes. Qdel distinguishes between queued requests and running requests. To delete a running request an explicit -k option is required. Therefore, a delete command of the form `qdel -k` *requestid* always flushes the desired request regardless of whether it is currently running.

The qdel command cannot be used to delete a request submitted to a remote batch queue once the request has been transmitted. To accomplish this, the remote shell command should be used. To delete a request on machine foo, for example, the command rsh foo /usr/alliant/qdel -k *requestid* could be used.

NOTES

- A batch request can only be deleted by the person submitting the request or by the superuser.

FILES

/usr/lib/batch/batchd	batch system daemon
/usr/spool/batch	batch spooling directory
/etc/qconf	batch system configuration file

SEE ALSO

qsub, qstat
Concentrix-2800 System Administration: batchd, The Batch System

qstat

show batch request status

The qstat program displays the status of current batch requests.

---

qstat [-al] [*queue*]

ARGUMENTS

-a shows all delayed and active requests.

-l uses the long format listing of requests, which displays the commandfile, priority, and submission time, in addition to the queue and request-id given with the short format.

queue lists only requests in the specified *queue*. By default, all requests in all queues are listed.

DESCRIPTION

Qstat provides a detailed listing of the batch system's queue, request, and device status. As with qdel, qstat does not provide transparent access to network batch status. In order to obtain remote batch status the remote shell command should be used. To obtain the batch system status on machine foo, for example, the command: rsh foo /usr/alliant/qstat -al could be used.

FILES

/usr/lib/batch/batchd	batch system daemon
/usr/spool/batch	batch spooling directory
/etc/qconf	batch system configuration file

SEE ALSO

qsub, qdel
Concentrix-2800 System Administration: batchd, The Batch System

qsub

submit batch request

The qsub command submits a batch request to the Alliant batch system.

---

qsub [*option...*] [*commandfile*]

ARGUMENTS

-a *time* runs the request after the specified *time*. Use this option to start large jobs at night when the computer is lightly loaded. For the format of *time*, see Time Format below.

-me sends mail to the user on the originating machine when the batch request has completed. If the -mu option is also present, mail is sent to the user specified by the -mu option instead.

-mu *username*
causes any mail concerning the request to be delivered to *username*. *Username* can be formatted as *user* or *user@machine*. If omitted, any mail concerning the request is sent to the invoker on the originating machine.

-o *outfile* redirects the standard output of the batch request to the named file.

-p *priority* assigns the specified *priority* to the request. Priorities range from 0 to 127 where 0 is the highest. The default is 64. Priorities serve only to order pending batch requests, there is no linkage with system scheduling priority.

-q *queue* specifies a queue for the request. The default queue is `batch`, other batch queues can be set up by the user (refer to qconf (Special Files)).

-r *title* assigns the specified *title* to the request. If this option is omitted, the title of the request defaults to the name of *commandfile*, if given, or to `STDIN`.

-s *shell* uses the specified *shell* to interpret *commandfile*. The default shell is specified by the environment variable `SHELL`. If that is undefined, the default shell is `/bin/sh`.

commandfile
specifies a batch command file. If omitted, the commands are read from the standard input until CTRL D (end-of-file) or CTRL B (abort) is received.

TIME FORMAT

The *time* character string consists of zero or more specifications of the following form:

tod time of day in the form *hh*:*mm*[:*ss*] [*meridian*] [*zone*]. If no meridian (am or pm) is specified, a 24-hour clock is used. A *tod* can be specified as just *hh* followed by a *meridian*.

date a specific month and day, and possibly a year. Acceptable formats are *mm*/*dd*[/*yy*] and *monthname dd*[, *yy*] If omitted, the year defaults to the current year; if a year is specified as a number less than 100, 1900 is added.

day a day of the week; the current day is used if appropriate. A *day* can be preceeded by a *number*, indicating which instance of that day is desired; the

default is 1. Negative *numbers* indicate times past. Some symbolic *numbers* are accepted: last, next, and the ordinals first through twelfth (second is ambiguous, and is not accepted as an ordinal number). The symbolic number `next` is equivalent to 2; thus, `next monday` refers not to the immediately coming Monday, but to the one a week later.

relative time
: a specification relative to the current time. The format is [*number*] *unit*; acceptable units are `year`, `month`, `fortnight`, `week`, `day`, `hour`, `minute`, and `second`.

The actual date is formed as follows: first, any absolute date and/or time is processed and converted. Using that time as the base, day-of-week specifications are added; last, relative specifications are used. If a date or day is specified, and no absolute or relative time is given, midnight is used. Finally, a correction is applied so that the correct hour of the day is produced after allowing for daylight savings time differences.

The time recognizer accepts most common abbreviations for days, months, etc.; in particular, it recognizes them with upper or lower case first letter, and recognizes three-letter abbreviations for any of them with or without a trailing period. Units, such as weeks, can be specified in the singular or plural. Timezone and meridian values can be in upper or lower case, and with or without periods. If you give more than one specification, quote the string.

DESCRIPTION

Batch requests submitted by the qsub command must be in the form of command shell scripts, either contained in a file or typed in via standard input from the user. When submitted, the user's environment, including exported shell variables and current working directory, are saved with the request and restored when the request is run. The only exception to this is that when a request is submitted to a remote batch queue, the working directory defaults to /tmp when the request is run on the remote machine. If the user wants the request to run in a particular directory on the remote machine a change directory (cd) command should be used in the shell command file. Once a request has been queued, it can be deleted with the qdel program, and the current status determined with the qstat program.

NOTES

- If a job submitted via qsub is running when the system crashes, the job is restarted automatically (by the batch daemon) when the system reboots. This is not always desirable. In cases where a batch job should NOT be restarted from the beginning without manual intervention, a workaround exists: submit a shell script which checks for the existence of some output file or lock file before starting the job.

EXAMPLES

```
qsub -a "3/1 03:22" foo          submit foo, run at 3:22 AM on March 1.
qsub -me foo                     submit foo, send mail upon completion.
qsub
make program >& make.errs
^D                               submit make command from standard input.
```

FILES

/usr/lib/batch/batchd	batch system daemon
/usr/spool/batch	batch spooling directory
/etc/qconf	batch system configuration file

SEE ALSO

qdel, qstat
Concentrix-2800 System Administration: batchd, The Batch System

quota

display disk usage and limits

---

quota [-v] [*user* ...]

ARGUMENTS

-v	display user's quotas on all mounted file systems where quotas exist
user	display quotas for the specified *user*

DESCRIPTION

The quota command displays a user's disk usage and limits. By default, the display is for you. Only the super-user can specify the *user* argument to view the limits of other users.

Quota without options displays only warnings about mounted file systems where usage is over quota. Remotely mounted file systems that are mounted with the "noquota" option are ignored. The -v option provides a full display: space usage, quota (soft limit), limit (hard limit), and time left (if over the soft limit); file usage, quota (soft limit), limit (hard limit), and time left (if over the soft limit). Zero indicates no limit.

EXAMPLES

```
% quota
Over disk quota on /usr, remove 10K within 7.0 days

% quota -v
Disk quotas for bob (uid 116):
Filesystem     usage  quota  limit    timeleft  files  quota  limit    timeleft
/usr              8K    10K    20K                  0      0      0
```

FILES

quotas	quota file at the file system root
/etc/mtab	list of currently mounted filesystems

SEE ALSO

Concentrix-2800 System Reference Manual: *Miscellaneous Routines, Special Files, and File Formats:* fstab
Concentrix-2800 System Reference Manual: *System Calls and C Library Routines:* quotactl, rquota
Concentrix-2800 System Administration: edquota, quotacheck, quotaon, quotaoff, repquota, rquotad

ranlib

convert archives to random-access libraries

The ranlib command converts each *archive* to a form that the loader can load more rapidly. Ranlib does this by adding a table of contents called __.SYMDEF (two underscores) to the beginning of the archive. Ranlib uses ar to reconstruct the archive, so that sufficient temporary file space must be available in the file system that contains the current directory.

---

ranlib [–t] *archive...*

ARGUMENTS

–t directs ranlib to only touch the archives, but not to modify them. Touching a file updates its modification time. This is useful with make, so that make doesn't complain about an out-of-date archive.

archive specifies an archive to convert.

NOTES

- Because generation of a library by ar and randomization of the library by ranlib are separate processes, phase errors are possible. The loader, ld, warns when the modification date of a library is more recent than the creation date of its dictionary; but this means that you get the warning even if you only copy the library.

- The FX/2800 archiver now uses System V archive library format to maintain all new libraries. Longnames are supported and the symbol index is now part of the archive. Old libraries are still maintained in the old (4.2BSD) format, and the archiver continues to accept archive files in the old format.

  It is not necessary to run ranlib on libraries after they are copied, if they use the new archive format. You must still use ranlib on old archives. If you use ranlib on a System V archive, the program recognizes that it is not necessary, and simply exits.

SEE ALSO

ld, ar, lorder

ratfor

"rational Fortran" preprocessor

The ratfor command invokes the Ratfor ("Rational Fortran") preprocessor.

---

ratfor [infile...] > *outfile*

ARGUMENTS

infile specifies an input file containing Ratfor source code.

outfile specifies an output file to contain Fortran source code.

DESCRIPTION

Ratfor provides control flow constructs essentially identical to those in C.

Statement Grouping

```
{ statement; statement; statement }
```

Decision Making

```
if (condition) statement [ else statement ]
switch (integer-value) {
        case integer:     statement
                .
                .
                .
        [ default: ]      statement
}
```

Loops

```
while (condition) statement
for (expression; condition; expression) statement
do limits statement
repeat statement [ until (condition) ]
break
next
```

Free Form Input

multiple statements/line; automatic continuation

Comments

```
# this is a comment
```

Translation of Relationals

>, >=, etc., become `.GT.`, `.GE.`, `etc.`

Return (expression)

returns expression to caller from function

Define

`define` *name replacement*

Include

`include` *filename*

SEE ALSO

Concentrix Programming Tools Guide: Ratfor

rcp

remote file copy

The rcp command copies files between machines.

---

rcp [-p] ***file1 file2***

rcp [-p] [-r] ***file... directory***

ARGUMENTS

file, file2, file, directory

are either remote file names of the form *rhost:path* or local file names containing no unquoted colons. If *path* is not a full path name, it is interpreted relative to your login directory on rhost.

A *path* on a remote host can be quoted using backslash (\), double quote ("), or single quote (') so that the metacharacters are interpreted remotely.

-p causes rcp to attempt to preserve (duplicate) in its copies the modification times and modes of the source files, ignoring the umask.

By default, the mode and owner of *file2* are preserved if it already existed; otherwise the mode of the source file modified by the umask on the destination host is used.

-r (recursive) if any of the source files are directories, rcp copies each subtree rooted at that name; in this case the destination must be a directory.

DESCRIPTION

Rcp does not prompt for passwords; your current local user name must exist on rhost and allow remote command execution via rsh.

Rcp handles third party copies, where neither source nor target files are on the current machine.

Hostnames can also take the form *username@hostname;filename* to use *username* rather than the current local user name on the remote host. rcp also supports Internet domain addressing of the remote host so that:

```
username@host.domain:filename
```

specifies the username to be used, the hostname, and the domain in which that host resides. Filenames that are not full pathnames will be interpreted relative to the home directory of the user named *username*, on the remote host.

The destination hostname may also take the form *hostname.username:filename* to support destination machines that are running older versions of rcp.

NOTES

- Rcp doesn't detect all cases where the target of a copy might be a file in cases where only a directory should be valid.
- Rcp *is* confused by any output generated by commands in a .login, .profile, or .cshrc file on the remote host.
- Rcp requires that the source host have permission to execute commands on the remote host when doing third-party copies.
- If you forget to quote metacharacters intended for the remote host you get an incomprehesible error message.
- Please note: rcp is meant to copy from one host to another; if by some chance you try to copy a file on top of itself, you will end up with a severely corrupted file (for example, if you executed the following command from host george: george% rcp testfile george:/usr/me/testfile). Remember where you are at all times (putting your hostname in your prompt helps with this)!

FILES

.chsrc
.login
.profile

SEE ALSO

cp, ftp, rsh, rlogin

rdist

remote file distribution program

Rdist is a program to maintain identical copies of files over multiple hosts.

---

rdist [-nqbRhivwy] [-f *distfile*] [-d *var=value*] [-m*host*] [*name* ...]

rdist [-nqbRhivwy] -c *name* ... [*login@*]*host*[*:dest*]

ARGUMENTS

-n Print the commands without executing them. This option is useful for debugging *distfile*.

-q Quiet mode. Files that are being modified are normally printed on standard output. The -q option suppresses this.

-b Binary comparison. Perform a binary comparison and update files if they differ rather than comparing dates and sizes.

-R Remove extraneous files. If a directory is being updated, any files that exist on the remote host that do not exist in the master directory are removed. This is useful for maintaining truly identical copies of directories.

-h Follow symbolic links. Copy the file that the link points to rather than the link itself.

-i Ignore unresolved links. Rdist will normally try to maintain the link structure of files being transferred and warn the user if all the links cannot be found.

-v Verify that the files are up to date on all the hosts. Any files that are out of date will be displayed but no files will be changed nor any mail sent.

-w Whole mode. The whole file name is appended to the destination directory name. Normally, only the last component of a name is used when renaming files. This will preserve the directory structure of the files being copied instead of flattening the directory structure. For example, renaming a list of files such as (dir1/f1 dir2/f2) to dir3 would create files dir3/dir1/f1 and dir3/dir2/f2 instead of dir3/f1 and dir3/f2.

-y Younger mode. Files are normally updated if their mtime and size (see stat(2)) disagree. The -y option causes rdist not to update files that are younger than the master copy. This can be used to prevent newer copies on other hosts from being replaced. A warning message is printed for files which are newer than the master copy.

-f name of file containing commands to direct the updating of files and/or directories. If *distfile* is '-', the standard input is used. If no -f option is present, the program looks first for *distfile*, then *Distfile* to use as the input.

-d Define *var* to have *value*. The -d option is used to define or override variable definitions in the *distfile*. Value can be the empty string, one name, or a list of names surrounded by parentheses and separated by tabs and/or spaces.

-m Limit which machines are to be updated. Multiple -m arguments can be given to limit updates to a subset of the hosts listed the *distfile*.

-c forces rdist to interpret the remaining arguments as a small *distfile*. The equivalent *distfile* is:

```
( name ... ) -> [login@]host
install [dest] ;
```

DESCRIPTION

Rdist is a program to maintain identical copies of files over multiple hosts. It preserves the owner, group, mode, and mtime of files if possible and can update programs that are executing.

Rdist reads commands from *distfile* to direct the updating of files and/or directories. If *distfile* is '-', the standard input is used. If no -f option is present, the program looks first for *distfile*, then *Distfile* to use as the input.

If no *names* are specified on the command line, rdist will update all of the files and directories listed in *distfile*. Otherwise, the argument is taken to be the name of a file to be updated or the label of a command to execute. If label and file names conflict, it is assumed to be a label. These may be used together to update specific files using specific commands.

Distfile contains a sequence of entries that specify the files to be copied, the destination hosts, and what operations to perform to do the updating. Each entry has one of the following formats:

```
<variable name> `=' <name list>

[label:] <source list> `->' <destination list> <command list>

[label:] <source list> `::' <timestamp file> <command list>
```

The first format is used for defining variables. The second format is used for distributing files to other hosts. The third format is used for making lists of files that have been changed since some given date.

The source list specifies a list of files and/or directories on the local host which are to be used as the master copy for distribution. The destination list is the list of hosts to which these files are to be copied. Each file in the source list is added to a list of changes if the file is out of date on the host which is being updated (second format) or the file is newer than the time stamp file (third format).

Labels are optional. They are used to identify a command for partial updates.

Newlines, tabs, and blanks are only used as separators and are otherwise ignored. Comments begin with '#' and end with a newline.

Variables to be expanded begin with '$' followed by one character or a name enclosed in curly braces (see the examples at the end).

The source and destination lists have one of the following format:

```
<name>

`(' <zero or more names separated by white-space> `)'
```

The shell meta-characters '[', ']', '{', '}', '*', and '?' are recognized and expanded (on the local host only) in the same way as csh(1). They can be escaped with a backslash. The '~' character is also expanded in the same way as csh but is expanded separately on the local and destination hosts. When the -w option is used with a file name that begins with '~', everything except the home directory is appended to the destination name. File names which do not begin with '/' or '~' use the destination user's home directory as the root directory for the rest of the file name.

The command list consists of zero or more commands of the following format.

```
`install' <options>  optdestname `;'

`notify'  <name list> `;'

`except'  <name list> `;'

`exceptpat' <pattern list>`;'

`special' <name list> string `;'
```

The install command is used to copy out of date files and/or directories. Each source file is copied to each host in the destination list. Directories are recursively copied in the same way. *Optdestname* is an optional parameter to rename files. If no install command appears in the command list or the destination name is not specified, the source file name is used.

Directories in the path name will be created if they do not exist on the remote host. To help prevent disasters, a non-empty directory on a target host will never be replaced with a regular file or a symbolic link. However, under the '-R' option a non-empty directory will be removed if the corresponding filename is completely absent on the master host.

The options are '-R', '-h', '-i', '-v', '-w', '-y', and '-b' and have the same semantics as options on the command line except they only apply to the files in the source list. The login name used on the destination host is the same as the local host unless the destination name is of the format "login@host".

The notify command is used to mail the list of files updated (and any errors that may have occurred) to the listed names. If no '@' appears in the name, the destination host is appended to the name (e.g., name1@host, name2@host, ...).

The except command is used to update all of the files in the source list except for the files listed in name list. This is usually used to copy everything in a directory except certain files.

The exceptpat command is like the except command except that pattern list is a list of regular expressions (see ed(1) for details). If one of the patterns matches some string

within a file name, that file will be ignored. Note that since '\' is a quote character, it must be doubled to become part of the regular expression. Variables are expanded in pattern list but not shell file pattern matching characters. To include a '$', it must be escaped with '\'.

The special command is used to specify sh(1) commands that are to be executed on the remote host after the file in name list is updated or installed. If the name list is omitted then the shell commands will be executed for every file updated or installed. The shell variable 'FILE' is set to the current filename before executing the commands in string. *String* starts and ends with a double quote (") and can cross multiple lines in *distfile*. Multiple commands to the shell should be separated by ';'. Commands are executed in the user's home directory on the host being updated. The special command can be used to rebuild private databases, etc. after a program has been updated.

NOTES

- Source files must reside on the local host where rdist is executed.
- There is no easy way to have a special command executed after all files in a directory have been updated.
- Variable expansion only works for name lists; there should be a general macro facility.
- Rdist aborts on files which have a negative *mtime* (before Jan 1, 1970).
- There should be a 'force' option to allow replacement of non-empty directories by regular files or symbolic links. A means of updating file modes and owners of otherwise identical files is also needed.

EXAMPLE

```
HOSTS = ( matisse root@arpa)

FILES = ( /bin /lib /usr/bin /usr/games
/usr/include/{*.h,{stand,sys,vax*,pascal,machine}/*.h}
/usr/lib /usr/man/man? /usr/ucb /usr/local/rdist )

EXLIB = ( Mail.rc aliases aliases.dir aliases.pag
crontab dshrc sendmail.cf sendmail.fc sendmail.hf
sendmail.st uucp vfont )

${FILES} -> ${HOSTS}
install -R ;
except /usr/lib/${EXLIB} ;
except /usr/games/lib ;
special /usr/lib/sendmail "/usr/lib/sendmail -bz" ;

srcs:
/usr/src/bin -> arpa
exceptpat ( \\.o\$ /SCCS\$ ) ;

IMAGEN = (ips dviimp catdvi)
```

```
imagen:
/usr/local/${IMAGEN} -> arpa
install /usr/local/lib ;
notify ralph ;

${FILES} :: stamp.cory
notify root@cory ;
```

DIAGNOSTICS

A complaint about mismatch of rdist version numbers may really stem from a problem starting your shell, e.g., you are in too many groups.

FILES

distfile	input command file
/tmp/rdist*	temporary file for update lists

SEE ALSO

sh, csh
Concentrix System Reference Manual: System Calls and C Library Routines: stat

read

read one line from input and assign words to variable (Bourne shell)

Read reads one line from standard input and assigns successive words to variables *name* in order. Leftover words go to the last variable.

---

read [*name* ...]

NOTES

- The return code is 0 unless the end-of-file is encountered.
- The read command is a Bourne shell special command.

readonly

given names are marked readonly (Bourne shell)

Readonly marks given names as readonly; subsequent assignment cannot change these values. If no arguments are given, readonly prints a list of all readonly names.

---

readonly [*name* ...]

NOTES

- The readonly command is a Bourne shell special command.

refer

find/insert literature references (unsupported)

The refer command invokes the Refer preprocessor for nroff or troff that finds and formats references for footnotes or endnotes. It is also the base for a series of programs designed to index, search, sort, and print stand-alone bibliographies, or other data entered in the appropriate form.

---

refer [*option...*] [*file...*]

ARGUMENTS

-a*n* reverses the first *n* author names (Jones, J. A. instead of J. A. Jones). If *n* is omitted all author names are reversed.

-b bare mode: do not put any flags in text (neither numbers nor labels).

-c*keys* capitalizes (with CAPS SMALL CAPS) the fields whose key-letters are in *keys*.

-e instead of leaving the references where encountered, accumulate them until a sequence of the form:

```
.[
$LIST$
.]
```

is encountered, and then write out all references collected so far. Collapse references to same source.

-f*n* sets the footnote number to *n* instead of the default of one. With labels rather than numbers, this option is a no-op.

-k*x* instead of numbering references, uses labels as specified in a reference data line beginning *%x*; by default *x* is L.

-l*m,n* instead of numbering references, uses labels made from the senior author's last name and the year of publication. Only the first *m* letters of the last name and the last *n* digits of the date are used. If either *m* or *n* is omitted the entire name or date respectively is used.

-n do not search the default file `/usr/dict/papers/Ind`. If there is a REFER environment variable, the specified file is searched instead of the default file; in this case the -n option has no effect.

-p *bib* specifies a file of references to be searched. The default file is searched last.

-s*keys* sorts references by fields whose key-letters are in the *keys* string; permute reference numbers in text accordingly. Implies -e. The key-letters in *keys* can be

ollowed by a number to indicate how many such fields are used, with + taken as a very large number. The default is AD, which sorts on the senior author and then date; to sort, for example, on all authors and then title, use -sA+T.

-B*l.m* bibliography mode. Takes a file composed of records separated by blank lines, and turns them into troff input. Label *l* turns into the macro *.m* with *l* defaulting to %X and *.m* defaulting to .AP (annotation rpara).

-P places punctuation marks .,:;?! after the reference signal, rather than before. (Periods and commas used to be done with strings.)

-S produces references in the Natural or Social Science format.

file (if not supplied, the standard input) is copied to the standard output, except for lines between .[and .] delimiters, which are assumed to contain keywords, and are replaced by information from the bibliographic database.

DESCRIPTION

Given an incomplete citation with sufficiently precise keywords, refer searches a bibliographic database for references containing these keywords anywhere in the title, author, journal, etc. The user can also search different databases, override particular fields, or add new fields. The reference data, from whatever source, are assigned to a set of *troff* strings. Macro packages such as ms print the finished reference text from these strings. By default references are flagged by footnote numbers.

To use your own references, put them in the format described below. They can be searched more rapidly by running indxbib on them before using refer; failure to index results in a linear search. When refer is used with the eqn, neqn or tbl preprocessors refer should be first, to minimize the volume of data passed through pipes.

The refer preprocessor and associated programs expect input from a file of references composed of records separated by blank lines. A record is a set of lines (fields), each containing one kind of information. Fields start on a line beginning with a per cent sign, followed by a key-letter, then a blank, and finally the contents of the field, and continue until the next line starting with a per cent sign. The output ordering and formatting of fields is controlled by the macros specified for nroff/troff (for footnotes and endnotes) or roffbib (for stand-alone bibliographies). For a list of the most common key-letters and their corresponding fields, see addbib. An example of a *refer* entry is given below.

EXAMPLE

```
%A      M. E. Lesk
%T      Some Applications of Inverted Indexes on the UNIX System
%B      UNIX Programmer's Manual
%V      2b
%I      Bell Laboratories
%C      Murray Hill, NJ
%D      1978
```

NOTES

- Blank spaces at the end of lines in bibliography fields cause the records to sort and reverse incorrectly.
- Sorting large numbers of references causes a core dump.

FILES

/usr/dict/papers	directory of default publication lists
/usr/lib/refer	directory of companion programs

SEE ALSO

addbib, sortbib, roffbib, indxbib, lookbib

rehash

recompute path table (C shell)

The rehash command recomputes the internal hash table of the contents of the directories in the *path* variable. Use rehash if new commands are added to the directories in your path while you are logged in. For example, if you add commands to one of your own directories or if the system administrator changes the contents of a system directory, a rehash would be required.

---

rehash

SEE ALSO

hashstat, unhash

repeat

execute command repetitively (C shell)

The repeat command executes a specified command a specified number of times.

---

repeat *count command*

ARGUMENTS

count specifies the number of times to execute *command*.

command cannot be a pipeline, a background command, commands separated by semicolons, nor another control command.

NOTES

- I/O redirections occur exactly once, even if *count* is 0.

reset

reset the teletype bits to a sensible state

The reset command has been merged with the tset command. Typing **reset**, however, still performs the function of resetting the terminal to a sensible state.

reten

retension tape cartridge

The reten command retensions tape cartridges by making a full, high-speed pass in each direction, ensuring uniform tape tension. Reten has no effect whatsoever on the contents of the cartridge.

---

reten [*device*]

ARGUMENTS

device optionally specifies a device name (the default is `/dev/rst00` for the streaming tape drive.)

DESCRIPTION

In normal use, cartridge tapes can develop uneven tension due to constant forward-backward (non-streaming) motion. A badly wound tape can produce I/O errors.

NOTES

- Reten cannot be killed, not even by `kill -KILL`.

rev

reverse lines of a file

The rev command copies the named files to the standard output, reversing the order of characters in every line.

---

rev [*file...*]

ARGUMENTS

file if not specified, the standard input is copied.

rlogin

remote login

Rlogin connects your terminal to a remote host system.

---

rlogin [**-ec**] [**-8**] [**-L**] [**-l** *username*] *hostname*

ARGUMENTS

-ec specifies an escape character (default is tilde). There is no space separating the option and argument character.

-8 provides an eight-bit path to the remote host with no interpretation. This allows programs that use CTRL S and CTRL Q as input (such as EMACS) to work correctly over an rlogin connection.

-L allows the rlogin session to be run in litout mode.

-l *username* specifies the remote user. If the originating user is not equivalent to the remote user, a login and password are prompted for on the remote machine as in login.

hostname Hostnames are listed in the *hosts* database, which may be contained in the /etc/hosts file, the NIS *hosts* database, the Internet domain name server, or a combination of these. Each host has one official name (the first name in the database entry), and optionally one or more nicknames. Either official hostnanes or nicknames may be specified in *hostname*.

DESCRIPTION

Each host has a file /etc/hosts.equiv which contains a list of *rhosts* with which it shares account names. (The host names must be the standard names as described in rsh.) When you rlogin as the same user on an equivalent host, you don't need to give a password.

Each user may also have a private equivalence list in a file .rhosts in his login directory. Each line in this file should contain an rhost and a username separated by a space, giving additional cases where logins without passwords are to be permitted. If the originating user is not equivalent to the remote user, then a login and password will be prompted for on the remote machine as in login.

To avoid some security problems, the .rhosts file must be owned by either the remote user or root. Moreover, the .rhosts file must only be writeable by the owner. Root will always be prompted for a password when executing rlogin.

The remote terminal type is the same as your local terminal type (as given in your environment TERM variable). The terminal or window size is also copied to the remote

system if the server supports the option, and changes in size are reflected as well. All echoing takes place at the remote site, so that (except for delays) the rlogin is transparent. Flow control via CTRL S and CTRL Q and flushing of input and output on interrupts are handled properly.

A tilde (~) appearing as the first character of a line is an escape signal; the following are recognized:

~. drops the connection and exits.

~CTRL Z suspends the rlogin process (only available with job control).

~CTRL Y suspends the rlogin process (only available with job control) but leaves the process that reads from the remote host alone.

NOTES

- If the directory /usr/hosts is set up correctly and in your search path, the `rlogin` command itself can be omitted. For example, if the directory contains a file kryptn, then:

  ```
  % kryptn
  ```

  is equivalent to:

  ```
  % rlogin kryptn
  ```

- More of the environment should be propagated.
- This implementation can only use the TCP network service.

FILES

/usr/hosts/*	for hostname version of the command.
/etc/hosts.equiv	list of trusted hostnames with shared usernames
~/.rhosts	private list of trusted hostname/username combinations

SEE ALSO

ypcat, rsh, tty
Concentrix-2800 System Administration: MAKEHOSTS, named
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: hosts

rm

remove (unlink) files

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

---

rm [-f] [-r] [-i] [-] *file...*

ARGUMENTS

-f forces removal. No questions are asked and no errors are reported.

-r recursively deletes the entire contents of the specified directory, and the directory itself.

-i interactively removes files. Rm asks whether to delete each file, and, under -r, whether to examine each directory.

- indicates that all the arguments following it are to be treated as file names. This allows the specification of file names starting with a hyphen.

DESCRIPTION

If a file has no write permission and the standard input is a terminal, its permissions are displayed and a line is read from the standard input. If that line begins with y the file is deleted, otherwise the file remains.

If a designated file is a directory, an error comment is displayed unless the optional argument -r has been used.

SEE ALSO

rmdir
Concentrix-2800 System Reference Manual: System Calls and C Library Routines: unlink, rmdir

rmail

handle remote mail received via uucp

The rmail command interprets incoming mail received via *uucp*, collapsing `From` lines in the form generated by binmail into a single line of the form "return-path!sender", and passing the processed mail on to sendmail. Rmail is explicitly designed for use with uucp and sendmail.

---

rmail *user...*

ARGUMENTS

user is the name of a user on a remote system.

SEE ALSO

binmail, uucp
Concentrix-2800 System Administration: sendmail

rmdel

remove a delta from an SCCS file

The rmdel command removes the specified delta from each named SCCS file.

---

rmdel **-r***SID file...*

ARGUMENTS

-r*SID* specifies the delta to be removed, which must be the newest (most recent) delta in its branch in the delta chain of each named SCCS file. In addition, the SID specified must *not* be that of a version being edited for the purpose of making a delta (i. e., if a *p-file* (see get) exists for the named SCCS file, the SID specified must *not* appear in any entry of the *p-file*).

file specifies the files to be processed. If a directory is named, rmdel behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored. If a hyphen is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file to be processed; non-SCCS files and unreadable files are silently ignored.

DESCRIPTION

The exact permissions necessary to remove a delta are documented in the *Concentrix Programming Tools Guide: Source Code Control System – System V Interface*. Simply stated:

- If you make a delta you can remove it; or
- If you own the file and directory you can remove a delta.

DIAGNOSTICS

- Use helpsccs for explanations.

FILES

x-file	(see delta)
z-file	(see delta)

SEE ALSO

sccs, delta, get, help, prs
Concentrix Programming Tools Guide: Source Code Control System – System V Interface
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: sccsfile

rmdir

remove (unlink) directories

The rmdir command removes entries for the named directories, which must be empty.

---

rmdir *dir...*

ARGUMENTS

dir specifies the name of a directory to remove.

SEE ALSO

rm

rpcgen

an RPC protocol compiler

Rpcgen is a tool that generates C code to implement an RPC protocol. The input to rpcgen is a language similar to C known as RPC. (Remote Procedure Call Language).

---

rpcgen *infile*
rpcgen [**-D***name*[=*value*]] [**-I** [**-K***seconds*]] [**-L**] [**-T**] *infile*
rpcgen -c -h -l -m -t [**-o** *outfile*] [*infile*]
rpcgen [**-s** *transport*]... [**-o** *outfile*] [*infile*]

ARGUMENTS

-c Compile XDR routines.

-D*name* [=*value*]]

Define a symbol *name*. Equivalent to the **#define** directive in the source. If no *value* is given, *name* is defined as 1. This option may be called more than once.

-h Compile C data-definitions (a header file). The **-T** option can be used in conjunction to produce a header file that supports RPC dispatch tables.

-I compile support for inetd in the server side stubs. Such stubs can be self-started or can be started by inetd. When the server is self-started, it backgrounds itself by default. A special define symbol RPC_SVC_FG can be used to run the server process in foreground, or alternately, the user may just compile it without the -I option. If no client requests are pending, the inetd servers exit after 120 seconds (default). The default can be changed with the -K option. All error messages from intetd servers are always logged in with syslog.

-K*seconds* If the server was started by inetd, specify the time in seconds after which the server should exit if there is no further activity. This option is useful for customization. If *seconds* is zero, the server exits after serving that given request. If *seconds* is -1, the server hangs around forever, once started by inetd. This option is only valid with the **-I** option.

-l Compile into client-side stubs.

-L When the servers are started in the foreground, use syslog to log the server errors instead of printing them on the standard error.

-m Compile into server-side stubs, but do not generate a "main" routine. This option is useful for doing callback-routines and for people who need to write their own "main" routine to do initialization. For inetd support, they

should be compiled with the -l option. In such cases, it defines two global variables: _rpcpmstart and _rpcfdtype. The value of _rpcpmstart should be 1 or 0 depending upon whether it was started by inetd or not. The value of _rpcfdtype should be SOCK_STREAM or SOCK_DGRAM depending upon the type of the connection.

-o *outfile* Specify the name of the output file. If none is specified, standard output is used (-c, -h, -l,-m, -s and -t modes only).

-s *transport*

Compile into server-side stubs, using the the given transport. The supported transports are udp and tcp. This option may be invoked more than once so as to compile a server that serves multiple transports.

-t Compile into RPC dispatch table.

-T Generate the code to support RPC dispatch tables.

The lowercase options (-c, -h, -l, -m, -s, and -t) are used exclusively to generate a particular type of file, while the uppercase options (-D, -I, -L, and -T) are global and can be used with the other options.

DESCRIPTION

Rpcgen is a tool that generates C code to implement an RPC protocol. The input to rpcgen is a language similar to C known as RPC (Remote Procedure Call Language). Information about the syntax of RPC Language is available in the "rpcgen Programming Guide" chapter in the *Concentrix NFS User's Guide*.

rpcgen is normally used as in the first synopsis where it takes an input file and generates four output files. If the infile is named proto.x ,then rpcgen generates a header file in proto.h , XDR routines in proto_xdr.c, server-side stubs in proto_svc.c, and client-side stubs in proto_clnt.c .

The second synopsis provides special features which allow for the creation of more sophisticated RPC servers. These features include support for RPC dispatch tables, and user-provided #defines. The entries in the RPC dispatch table contain:

- pointers to the service routine corresponding to that procedure
- a pointer to the input and output arguments
- the size of these routines

A server can use the dispatch table to check authorization and then to execute the service routine; a client library may use it to deal with the details of storage management and XDR data conversion.

The other synopses shown above are used when one does not want to generate all the output files, but only a particular one. Their usage is described in the EXAMPLES section below.

The C-preprocessor, cpp, is run on all input files before they are actually interpreted by rpcgen, so all the cpp directives are legal within an rpcgen input file. For each type of output file, rpcgen defines a special cpp symbol for use by the rpcgen programmer:

RPC_HDR	defined when compiling into header files
RPC_XDR	defined when compiling into XDR routines
RPC_SVC	defined when compiling into server-side stubs
RPC_CLNT	defined when compiling into client-side stubs
RPC_TBL	defines when compiling into RPC dispatch tables

In addition, rpcgen does a little preprocessing of its own. Any line beginning with ' % ' is passed directly into the output file, uninterpreted by rpcgen. You can customize some of your XDR routines by leaving those data types undefined. For every data type that is undefined, rpcgen assumes that a routine exists with the name xdr_ prepended to the name of the undefined type.

EXAMPLES

The following example generates all the five files: prot.h, prot_clnt.c, prot_svc.c, prot_xdr.c, and prot_tbl.i. The server error messages are logged, instead of being sent to the standard error.

```
example% rpcgen -LT prot.x
```

The following example generates prot.h, prot_clnt.c, prot_svc.c, and prot_xdr.c. prot_svc.c supports server invocation by inetd. If the server is started by inetd, it exits after 20 seconds of inactivity.

```
example% rpcgen -I -K20 prot.x
```

The following example sends the header file (with support for dispatch tables) on the standard output.

```
example% rpcgen -hT prot.x
```

The following example sends the server-side stubs file for the transport tcp on the standard output.

```
example% rpcgen -s tcp prot.x
```

NOTES

- The RPC language does not support nesting. As a workaround, structures can be declared at top-level, and their name used inside other structures in order to achieve the same effect.
- Name clashes can occur when using program definitions, since the apparent scoping does not really apply. Most of these can be avoided by giving unique names for programs, versions, procedures, and types.

SEE ALSO

cpp
Concentrix-2800 System Administation: inetd
Concentrix-2800 System Reference: System Calls and C Routines: rpc
Concentrix Network File System User's Guide, chapter on "rpcgen Programming Guide"

roffbib

display bibliographic database (unsupported)

The roffbib command displays all records in a bibliographic database, in bibliography format rather than as footnotes or endnotes.

---

roffbib [*option...*] [*file...*]

ARGUMENTS

Roffbib accepts most of the options understood by nroff, most importantly the –T option to specify terminal type.

–x suppresses the display of %X abstracts.

–m *mac* specifies a set of user-defined macros to replace the ones defined in /usr/lib/tmac/tmac.bib. There must be a space between the –m and the macro filename.

–V sends output to the Versatec.

–Q queues output for the phototypesetter.

DESCRIPTION

Generally, roffbib is used in conjunction with sortbib:

```
sortbib database | roffbib
```

If abstracts or comments are entered following the %X field key, roffbib formats them into rparas for an annotated bibliography. Several %X fields can be given if several annotation rparas are desired.

Four command-line registers control formatting style of the bibliography, much like the number registers of ms. The command-line argument –rN1 numbers the references starting at one. The option –rV2 double spaces the bibliography, while –rV1 double spaces references but single spaces annotation rparas. The line length can be changed from the default 6.5 inches to 6 inches with the –rL6i argument, and the page offset can be set from the default of 0 to one inch by specifying –rO1i (capital O, not zero). Note: with the –V and –Q options the default page offset is already one inch.

NOTES

- Users have to rewrite macros to create customized formats.

FILES

/usr/lib/tmac/tmac.bib file of macros used by nroff/troff

SEE ALSO

refer, addbib, sortbib, indxbib, lookbib

rsh

remote shell

The rsh command connects to the specified host, and executes a specified command. Rsh copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit and terminate signals are propagated to the remote command; rsh normally terminates when the remote command does.

---

rsh *host* [**-l** *username*] [**-n**] *command*

ARGUMENTS

host specifies the name of the remote host. Host names are given in the file /etc/hosts. Each host has one standard name (the first name given in the file), which is rather long and unambiguous, and optionally one or more nicknames.

-l *username* must be equivalent (in the sense of rlogin) to the originating account; no provision is made for specifying a password with a command. *Rsh* uses your local username by default.

-n *command* specifies a single command to execute. If you omit *command*, you are logged in on the remote host using rlogin.

DESCRIPTION

Shell metacharacters that are not quoted are interpreted on local machine, while quoted metacharacters are interpreted on the remote machine. Thus the command

```
rsh otherhost cat remotefile >> localfile
```

appends the remote file *remotefile* to the localfile *localfile, while*

```
rsh otherhost cat remotefile ">>" otherremotefile
```

appends *remotefile* to *otherremotefile*.

NOTES

- If you are using csh and put a rsh in the background without redirecting its input away from the terminal, it blocks even if no reads are posted by the remote command. If no input is desired you should redirect the input of rsh to /dev/null using the -n option.
- You cannot run an interactive command (like rogue or vi); use rlogin.

- Stop signals stop the local rsh process only.
- If the directory /usr/hosts is set up correctly and in your search path, the `rsh` command itself can be omitted. For example, if the directory contains a file `kryptn`, then:

```
% kryptn pwd
```

is equivalent to:

```
% rsh kryptn pwd
```

FILES

/etc/hosts
/usr/hosts/*

SEE ALSO

rlogin
Concentrix-2800 System Administration: MAKEHOSTS

rt

read magnetic tape and print record sizes

The rt command reads a magnetic tape and prints out the size of each record read. Normally, rt is used when the record format is unknown.

---

rt *device*

ARGUMENTS

device specifies a device name (normally /dev/rxt00h for the 6250 bpi 1/2-inch magnetic tape drive.)

DESCRIPTION

Rt tries to read 60 Kbyte records and prints the size returned by the read system call. A record size of zero indicates an end-of-file mark (magnetic tapes can contain many end-of-file marks). Rt continues to read records until explicitly terminated by an interrupt signal or until the tape runs out.

rup

show host status of local machines (RPC version)

Rup gives a status similar to uptime for remote machines; it broadcasts on the local network, and displays the responses it receives.

---

rup [-h] [-l] [-t] [-b | *host* ...]

ARGUMENTS

-h sort the display alphabetically by host name

-l sort the display by load average

-t sort the display by up time

-b | *host*... if -b, broadcast a query to the whole network. If *host*, query list of hosts. User must type either a list of hostnames, or -b.

DESCRIPTION

Normally, the listing of the status is in the order that responses are received, but this order can be changed by specifying one of the options listed below.

A remote host will only respond if it is running the rstatd daemon, which is normally started up from inetd (See *Concentrix System Administration* or *Concentrix-2800 System Administation)*.

NOTES

- Broadcasting does not work through gateways.

FILES

/etc/servers

SEE ALSO

ruptime
Concentrix-2800 System Administration: inetd, rstatd

ruptime

show host status of local machines

The ruptime command gives a status line like uptime for each machine on the local network; these are formed from packets broadcast by each host on the network once a minute.

---

ruptime [-a] [-r] [-l] [-t] [-u]

ARGUMENTS

-a shows all users. Otherwise, users idle an hour or more are not counted.

-r reverses the sort order.

-l sorts by load average.

-t sorts by uptime.

-u sorts by number of users.

DESCRIPTION

Normally, the listing is sorted by host name.

Machines for which no status report has been received for eleven minutes are shown as being down.

FILES

/usr/spool/rwho/whod.* data files

SEE ALSO

rwho

rusers

who's logged in on local machines (RPC version)

---

rusers [-a] [-h] [-i] [-l] [-u] [-b | *host* ...]

ARGUMENTS

-a gives a report for a machine even if no users are logged on.

-h sort alphabetically by host name.

-i sort by idle time.

-l Give a longer listing in the style of who*(1)*.

-u sort by number of users.

-b | *host*... if -b, broadcast a query to the whole network. If *host*, query list of hosts. User must either use -b, or type in desired list of hostnames.

DESCRIPTION

The rusers command produces output similar to users and who, but for remote machines. It broadcasts on the local network, and prints the responses it receives. Normally, the listing is in the order that responses are received, but this order can be changed by specifying one of the options listed below. When host arguments are given, rather than broadcasting rusers will only query the list of specified hosts.

The default is to print out a listing in the style of users with one line per machine. When the -l flag is given, a rwho style listing is used. In addition, if a user hasn't typed to the system for a minute or more, the idle time is reported.

A remote host will only respond if it is running the rusersd daemon, which is normally started up from inetd.

NOTES

- Broadcasting does not work through gateways.

FILES

/etc/inetd.conf

SEE ALSO

rwho
Concentrix-2800 System Administration: inetd, rusersd

rwall

write to all users over a network

The rwall command sends a message, preceded by the line `Broadcast Message ...`, to all users logged in on the specified host machines.

---

rwall *host1 host2*

rwall **-n** *netgroup1 netgroup2...*

rwall **-h** *host* **-n** *netgroup*

ARGUMENTS

host	sends to the specified host.
-n *netgroup*	sends to the specified network groups (defined in *netgroup*).
-h *host*	sends to the specified host.

DESCRIPTION

Rwall reads a message from standard input until end-of-file. It then sends this message, preceded by the line "Broadcast Message ...", to all users logged in on the specified host machines. With the -n option, it sends to the specified network groups, which are defined in netgroup(5).

A machine can only receive the message if it is running rwalld, which is normally started up from `/etc/servers` by the daemon inetd.

NOTES

- The timeout is fairly short in order to be able to send to a large group of machines (some of which may be down) in a reasonable amount of time. Thus the message may not get through to a heavily loaded machine.

FILES

/etc/servers

SEE ALSO

wall
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: netgroup
Concentrix-2800 System Administration: inetd, rwalld, shutdown

rwho

show who's logged in on local machines

The rwho command produces output similar to who, but for all machines on the local network. If no report has been received from a machine for five minutes, rwho assumes the machine is down and does not report users last known to be logged into that machine.

---

rwho [-a]

ARGUMENTS

-a shows all users. If a user hasn't typed to the system for a minute or more, *rwho* reports his idle time. By default, a user who hasn't typed to the system for an hour or more is omitted from the output.

NOTES

- Rwho is unwieldy when the number of machines on the local net is large.

FILES

/usr/spool/rwho/whod.* information about other machines

SEE ALSO

ruptime
Concentrix-2800 System Administration: rwhod

scc

Scalar/C compiler command

Processes a Scalar/C program.

---

scc [*option* [*option*] ...] [*input-file-name* [*input-file-name*] ...]

DESCRIPTION

The scc command compiles a Scalar/C program. The following optimizations are not available with Scalar/C (but are available with FX/C-2800): vectorization, concurrency, pipelining, and dual instruction mode.

Processing occurs in several phases:

- scc0 – The Scalar/C preprocessor: processes #define, #include, and other preprocessor directives.
- scc1 – The compiler proper: processes the source code, performing any allowed global optimizations and inline expansions.
- ld – The link editor: processes object code.

By default, the scc command compiles C source files into object files and links the object files into an executable file named a.out. An input file can contain source program units or object programs. The name of a C source file must end with a period and the letter c. Files ending with a period and the letter s are assumed to be assembler files and are assembled into object files. All other files are assumed to be object files or libraries; see the ar and ranlib commands in the system documentation. The output object file names are the same as the source file names except that the last letter is o instead of c.

Error and warning messages are written to the standard error file.

If the link step is invoked, ld is called with command line arguments based on the arguments to the scc command: for each source file specified to scc, the corresponding object file is specified to ld; each scc argument is removed; each unrecognized argument is passed to ld; each object file specified to scc is specified to ld. The order of these arguments and file specifications is preserved. The library option –lc is specified to ld after all user arguments and file specifications.

Arguments not accepted by the scc command are passed to the ld command. If the link step is not invoked (–c, –S, –syntax), those arguments are ignored.

Several options to the scc command request optimizations. Since optimized object code does not always correspond to its source code in a straightforward way, you may wish to compile without optimization during program development and debugging. Then when development is completed, modules can be recompiled using optimizations.

OPTIONS

-c Suppresses the link phase so that no executable file is produced, unless -o is also specified.

-C Prevents the preprocessor from omitting comments when -E is also specified.

-D*name*[=*value*]
Defines *name*, as if through the preprocessor directive #define; *value* defaults to 1.

-double
Forces double precision for intermediate floating point calculations.

-E Executes only the C preprocessor and writes the output to the standard output unit.

-g Includes symbol table information for debuggers in the object files and the executable file.

-i Suppresses informational messages.

-inline *fcnname*
Enables the inline expansion facility and requires inline expansion of the function *fcnname*; does not enable automatic inline expansion. The -inline option takes a single argument *fcnname*; to specify additional functions, use additional -inline options.

-I*directory*
Searches the specified directories for #include files. The specified directories are searched after the directory containing the source file with the #include directive and before the directories on the standard search list (/usr/include).

-l Generates a listing file (to standard output).

-li Generates a listing file (to standard output) that includes the text of files specified by the #include preprocessor directive.

-l*name*
Links using library /lib/*libname*.a.

-M Generates the list of files included by the compiler when trying to compile a source file. This option is very useful when generating a list of dependencies.

-o ***filename***
Names the executable file. By default, the executable file is named a.out.

-O [g][i]
Optimizes the generated code as specified by suboptions:

g Allows global optimization.

i Enables the inline expansion facility and initially allows automatic inline expansion.

Option -O (with no suboptions) is the same as -Ogi.

-OM[i][p]
Suppresses listing options as specified by suboptions:

i Suppresses inline expansion summaries

p Suppresses the text of the source code

Option -OM (with no suboptions) is the same as -OMils.

-pcc Enables some portable C compiler (the old C compiler) features that conflict with ANSI C. Ineffective with the current release.

-pg Produces a program that (at program execution time) monitors the calling of routines and writes a gmon.out file. The command gprof processes gmon.out files.

-S Produces assembler files. One assembler file is produced for each C source file. An assembler file has the same name as the C file except that the letter c at the end is changed to the letter s. Assembler files *cannot* be assembled. Suppresses production of object files unless -c or -o is specified; suppresses production of an executable file unless -o is specified.

-syntax
Performs syntactic and semantic analysis and issues messages. Suppresses production of object files unless -c or -o is specified; suppresses production of an executable file unless -o is specified.

-U*name*
Undefines *name* as if through the preprocessor directive #undef.

-v Displays the compiler version number. This option can be used without naming an input file.

-w Suppresses warning messages.

EXAMPLES

The following example compiles and links the file named hello.c, producing an object file named hello.o and an executable file named a.out.

```
% scc hello.c
```

The following example compiles and links the file named hello.c, producing a listing file named hello.list, an object file named hello.o, and an executable file named hello.

```
% scc -l -o hello hello.c > hello.list
```

The following example compiles and links the files named hello.c, sub1.c, and sub2.c, producing object files named hello.o, sub1.o, and sub2.o, and an executable file named hello.

```
% scc -o hello hello.c sub1.c sub2.c
```

The following example compiles the file named sub1.c, producing an object file named sub1.o.

```
% scc -c sub1.c
```

The following example compiles the files named hello.c and and sub2.c, producing object files named hello.o and sub2.o; and links the object files and sub1.o into an executable file named hello.

```
% scc -o hello hello.c sub1.o sub2.c
```

The following example compiles the file named hello.c but does not produce an object file or an executable file.

```
% scc -syntax hello.c
```

The following example compiles the file named hello.c into an assembler file named hello.s.

```
% scc -S hello.c
```

The following example compiles the file named hello.c with full optimization, producing a listing file named hello.list, an object file named hello.o, and and executable file named hello. The listing contains no optimization messages or summaries.

```
% scc -l -O -AS -OM -o hello hello.c > hello.list
```

FILES

file.c	C input file
file.s	Assembler file
file.o	Object file
a.out	Executable file
gmon.out	File produced for analysis by gprof
/bin/scc	Scalar/C compiler command
/bin/scco	Scalar/C preprocessor
/bin/scc1	Scalar/C compiler
/lib/crt0.o	Run-time start-off
/tmp/c?????	Temporary file
/usr/include	Standard directory for C #include files
/usr/include/math.h	Math definitions
/usr/include/sys/errno.h	Error names and numbers
/usr/lib/gcrt0.o	Start-off for gprof profiling
/usr/lib/libc_p.a	Profiling library (unshared)
/usr/lib/libc_u.a	Unshared library

SEE ALSO

ld
The C Programming Language (Kernighan and Ritchie), second edition

SCCS

Source Code Control System (Berkeley interface)

The sccs command invokes the Berkeley interface to the Source Code Control System.

---

sccs [-r] [-d*path*] [-p*path*] *command* [*flags*] [*args*]

It is important to understand the distinction between the SCCS program, and an interface (such as this one) to the program. This interface, called sccs, passes commands to the SCCS program.

All commands that you enter with the sccs interface begin with the keyword sccs. The sccs interface may take some flags, which are entered directly after the keyword sccs. In addition, the SCCS commands may take their own flags and arguments. These are entered after the particular command and are passed to the SCCS program.

Note that this Berkeley sccs interface is different from the System V SCCS interface. The latter allows you to pass commands (such as get and delta) directly to the SCCS program (without beginning the command with the keyword sccs).

FLAGS FOR THE BERKELEY SCCS INTERFACE

-r runs SCCS as the real user, rather than as whatever effective user SCCS is set to with set user id. Certain commands (such as admin) cannot be run by all users, since this would allow anyone to change the authorizations. These commands are always run as the real user.

-d*path* gives a root directory for the SCCS files. The default is the current directory.

-p*path* defines the pathname of the directory in which the SCCS files will be found. SCCS is the default.

The -p flag differs from the -d flag in that the -d argument is prepended to the entire pathname and the -p argument is inserted before the final component of the pathname. For example, sccs -d/x -py get a/b will convert to get /x/a/y/s.b.

The intent here is to create aliases such as alias syssccs sccs -d/usr/src which will be used as syssccs get cmd/who.c. Also, if the environment variable PROJECT is set, its value is used to determine the -d flag. If it begins with a slash, it is taken directly; otherwise, the home directory of a user of that name is examined for a subdirectory src or source. If such a directory is found, it is used.

SCCS COMMANDS

admin creates or sets parameters on s-files.

-d*flag* turns off (deletes) the *flag*.

-f*flag* turns on the *flag*.

-i*file* creates, using *file* as the initial contents.

-t*file* replaces the descriptive text in the s-file with the contents of *file*. If *file* is omitted, the text is deleted. Useful for storing documentation or "design and implementation" documents to insure they get distributed with the s-file.

-z rebuilds the checksum in case the file has been trashed.

check like info except that nothing is printed if nothing is being edited, and a non-zero exit status is returned if anything is being edited. The intent is to have this included in an "install" entry in a makefile to insure that everything is included into the SCCS file before a version is installed.

clean removes all files that can be regenerated from the s-file.

create creates an SCCS file, taking the initial contents from the file of the same name. Any flags to admin are accepted. If the creation is successful, the files are renamed with a comma on the front. These should be removed when you are convinced that the SCCS files have been created successfully.

diffs Gives a diff listing between the current version of the program(s) you have out for editing and the versions in SCCS format. The -r, -c, -i, -x, and -t flags are passed to get; the -l, -s, -e, -f, -h, and -b options are passed to diff. The -C flag is passed to diff as -c.

get gets files for compilation (not for editing). Id keywords are expanded.

-c*date* suppresses application of deltas created after *date*.

-d*SID* default SID to be used.

-i*list* specifies a list of deltas to include.

-k suppresses expansion of id keywords.

-m precedes each line with SID of creating delta.

-p sends to the standard output rather than to the actual file.

-r*SID* specifies the version to get.

-x*list* specifies a list of deltas to exclude.

delget does a delta followed by a get.

deledit does a delta followed by an edit.

delta merges a file gotten using edit back into the s-file and collects comments about why this delta was made.

edit gets files for editing. Id keywords are not expanded. Should be matched with a delta command.

-b creates a branch.

-d*SID* default SID to be used.

-i*list* specifies a list of deltas to include.

-r*SID* specifies the version to get. If *SID* specifies a release that does not yet exist, the highest numbered delta is retrieved and the new delta is numbered with *SID*.

-x*list* specifies a list of deltas to exclude.

fix This command essentially removes the named delta, but leaves you with a copy of the delta with the changes that were in it. It is useful for fixing small compiler bugs, etc. Since it doesn't leave audit trails, it should be used carefully. Must be followed by an -r flag.

info gives a list of all files being edited.

-b ignores branches.

-u[*user*] list only files being edited by *user* (default is yourself).

print This command is not implemented.

prs produces a report of changes.

-e prints (nearly) everything.

-t prints the descriptive text.

tell is the same as info, except that one line is produced per file being edited containing only the file name.

unedit removes a file that has been edited previously, without merging the changes into the s-file. This is the opposite of an edit or a get -e. It should be used with extreme caution, since any changes you made since the get will be irretrievably lost.

what finds and prints id keywords.

DESCRIPTION

This documentation describes the syntax and commands of the Berkeley SCCS interface. For tutorial information about using this interface, see the *Concentrix Programming Tools Guide*.

NOTES

Some of the commands may not take a directory specifier properly, and will work only in the default directory.

ID KEYWORDS

%Z%	expands to `@(#)` for the *what* command to find.
%M%	the current module name, *e.g.*, `prog.c`.
%I%	the highest SID applied.
%W%	a shorthand for `%Z%%M%` TAB `%I%`.
%G%	the date of the delta corresponding to the `%I%` keyword.
%R%	the current release number, *i.e.*, the first component of the `%I%` keyword.
%Y%	replaced by the value of the *t* flag (set by *admin*).

SEE ALSO

Concentrix Programming Tools Guide: Source Code Control System – Berkeley Interface
Concentrix Programming Tools Guide: Source Code Control System – System V Interface

sccsdiff

compare two versions of an SCCS file

The sccsdiff command compares two versions of an SCCS file and generates the differences between the two versions.

---

sccsdiff -r*SID1* -r*SID2* [-p] [-s*n*] *file...*

ARGUMENTS

-r*SID1* and *SID2* specify the deltas of an SCCS file that are to be compared.

-p pipes output for each file through pr.

-s*n* specifies the file segment size that bdiff passes to diff. This is useful when diff fails due to a high system load.

file specifies a file to be processed. Any number of SCCS files can be specified, but arguments apply to all files.

DIAGNOSTICS

- *file*: `No differences` If the two versions are the same. Use help for explanations.

FILES

/tmp/get????? Temporary files

SEE ALSO

sccs, bdiff, get, help, pr
Concentrix Programming Tools Guide: Source Code Control System – System V Interface.

script

make typescript of terminal session

The script command creates a forked shell and makes a typescript of everything displayed on your terminal. The script ends when the forked shell exits.

---

script [-a] [*file*]

ARGUMENTS

-a appends the typescript to the specified *file*.

file specifies the name of a file to contain the typescript. The default is `typescript`.

DESCRIPTION

This program is useful when using a video terminal and a hard copy record of the dialog is desired, as for a student handing in a program that was developed on a video terminal when hard-copy terminals are in short supply.

NOTES

- Script places everything, including rubouts, in the log file, which is not what a naive user might expect.

sed

stream editor

The **sed** command copies the named files to the standard output, edited according to a script of commands.

---

sed [-n] [-e *script*] [-f *sfile*] [*file...*]

ARGUMENTS

These options accumulate.

-n suppresses the default output.

-e *script* specifies a quoted editing script. If there is just one **-e** option and no **-f**'s, the **-e** option can be omitted.

-f *sfile* causes the script to be taken from file *sfile*.

file specifies an input file.

DESCRIPTION

A script consists of editing commands, one per line, of the following form:

[address [, address]] function [argument...]

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

An *address* is either a decimal number that counts input lines cumulatively across files, a '$' that addresses the last line of input, or a context address, /regular expression/, in the style of *ed* modified thus:

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

A command line with no addresses selects every pattern space.

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

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

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

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

An argument denoted *text* consists of one or more lines, all but the last of which end with backslash to hide the newline. Backslashes in text are treated like backslashes in the replacement string of an s command, and can be used to protect initial blanks and tabs against the stripping that is done on every script line.

An argument denoted *rfile* or *wfile* must terminate the command line and must be preceded by exactly one blank. Each *wfile* is created before processing begins. There can be at most 10 distinct *wfile* arguments.

(1)a\
text Append. Place *text* on the output before reading the next input line.

(2)b *label*
Branch to the colon command bearing the *label*. If *label* is empty, branch to the end of the script.

(2)c\
text Change. Delete the pattern space. With 0 or 1 address or at the end of a two-address range, place *text* on the output. Start the next cycle.

(2)d Delete the pattern space. Start the next cycle.

(2)D Delete the initial segment of the pattern space through the first newline. Start the next cycle.

(2)g Replace the contents of the pattern space by the contents of the hold space.

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

(2)h Replace the contents of the hold space by the contents of the pattern space.

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

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

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

(2)N Append the next line of input to the pattern space with an embedded newline. (The current line number changes.)

(2)p Print. Copy the pattern space to the standard output.

(2)P Copy the initial segment of the pattern space through the first newline to the standard output.

(1)q Quit. Branch to the end of the script. Do not start a new cycle.

(2)r *rfile*
Read the contents of *rfile*. Place them on the output before reading the next input line.

(2)s/*regular expression*/*replacement*/*flags*
Substitute the *replacement* string for instances of the *regular expression* in the pattern space. Any character can be used instead of slash. For a fuller description see *ed*. *Flags* is zero or more of

g Global. Substitute for all nonoverlapping instances of the *regular expression* rather than just the first one.

p Print the pattern space if a replacement was made.

w *wfile*
Write. Append the pattern space to *wfile* if a replacement was made.

(2)t *label*
Test. Branch to the colon command bearing the *label* if any substitutions have been made since the most recent reading of an input line or execution of a t command. If *label* is empty, branch to the end of the script.

(2)w *wfile*
Write. Append the pattern space to *wfile*.

(2)x Exchange the contents of the pattern and hold spaces.

(2)y/*string1*/*string2*/
Transform. Replace all occurrences of characters in *string1* with the corresponding character in *string2*. The lengths of *string1* and *string2* must be equal.

(2)! *function*
Don't. Apply the *function* (or group, if *function* is left brace) only to lines *not* selected by the address(es).

(0): *label*
This command does nothing; it bears a *label* for b and t commands to branch to.

(1)= Place the current line number on the standard output as a line.

(2){ Execute the following commands through a matching right brace only when the pattern space is selected.

(0) An empty command is ignored.

SEE ALSO

ed, grep, awk, lex
Concentrix Programming Tools Guide: SED – A Non-Interactive Text Editor

sendbug

bug reporting utility

Sendbug is a utility that helps you prepare a problem report to be sent to Alliant Customer Services. The report is filled in by answering a series of multiple choice, yes-no, and fill-in-the-blanks questions. A text editor is used to submit the descriptive parts of the report. When complete, sendbug mails the report to Alliant and files a copy for the reporter.

---

sendbug

DESCRIPTION

After a few seconds, questions and directions appear on the screen to guide you through the report. Respond to multiple choice questions with a number followed by RETURN. Respond to yes-no questions with y or n followed by RETURN. Respond to fill-in-the-blank questions only in the area delimited by bold dashes and terminate the response with RETURN. Some of the questions are:

- Author

  The author field is filled in automatically with `'hostname'!'loginname'`. To provide a different response, wait for the following question:

  ```
  Is the header for the report correct (Y or N)?
  ```

  Answer n and select the author field to be updated.

- Product Revision

  The revision number can be found on the latest copy of the release notice that was shipped with the last release of the product. Please be accurate.

- Machine Description

  The preferred response is the serial number of the machine on which the problem occurred. If the serial number is not available, provide a brief description of the machine, for example, `FX/8,6ce,2ip`. If the problem involves a device, include the device type in the description. In general, if the problem seems to involve a hardware component, include the component in the description.

- Description, Procedure to Repeat Problem

  These fields (filled in using a text editor) are limited to 2000 characters (25 lines of 80 characters). If you need more space, add an attachment to the report. Remember to overwrite the capitalized reminders.

FILES

/usr/alliant/sendbug
/usr/alliant/sbheader
/usr/alliant/prsformat

NOTES

- Sendbug uses vi as the default text editor. If you prefer another editor, such as Emacs, place the following line in `~/.login`:

  ```
  setenv EDITOR /usr/bin/emacs
  ```

  If you are a Bourne shell user, place the following lines in `~/.profile`:

  ```
  EDITOR=/usr/bin/emacs
  export EDITOR
  ```

- The file /usr/lib/aliases contains an alias that defines the Mail path that sendbug uses to call the Alliant Customer Services Support Center. For example:

  ```
  csbugtrack: alliant\!csbugtrack
  ```

- If it has not already been done, call the Alliant Customer Services Support Center to have an account established on its computer for your site so that sendbug can mail your bug reports via UUCP.

set

set local variable (C shell)

The **set** command declares and assigns a value to a user-defined variable that is local to the current C shell. Shell variables are arrays of strings. Those that have a value of more than one word display as a parenthesized word list.

---

set [*name* [= [*word*]]]

set *name*[*index*] [= [*word*]] (Note: the brackets around *index* are literals)

set *name* = (*wordlist*)

ARGUMENTS

With no arguments, **set** displays the values of all shell variables.

name is the name of the variable to set. Variable names consist of 1 to 20 characters chosen from the same set of characters as a filename (letters, underscore, and digits, beginning with a letter).

name[*index*] specifies an element of *name*. To access individual elements of an array, use *index* numbers (counting from one). To access a range of elements, use a pair of *index* numbers separated with a hyphen. If omitted, the second *index* number defaults to the last element.

word is the value to assign (one word). The default is the null string.

(*wordlist*) is the value to assign (two or more words enclosed in parentheses).

The arguments can be repeated to set multiple values in a single **set** command. Variable substitution occurs for all arguments before any assignment occurs.

SPECIAL VARIABLES

Some shell variables directly affect the behavior of the C shell and/or are predefined by the shell. Some of the special variables (**echo**, **ignoreeof**, and so forth) are boolean; the shell only recognizes if they are set or not. These descriptions are very brief. Refer to the *Concentrix System User's Guide* for details.

argv contains command line arguments.

cdpath specifies the search path for commands that change the current directory.

cwd contains the full pathname of the current directory.

echo echoes each command and its arguments just before execution.

histchars changes the characters used in history substitution.

history controls the size of the history list.

home contains your home directory.

ignoreeof prevents exit on end-of-file from input devices that are terminals.

mail specifies the files where the shell checks for mail.

noclobber restricts output redirection to prevent accidental destruction of files and appendages to nonexistant files.

noglob suppresses filename substitution.

nonomatch suppresses errors from nonmatching filename substitutions.

notify notifies you asynchronously of job completions.

path specifies the search path for command execution and corresponds to the environment variable PATH.

prompt specifies the string printed before each command.

savehist controls the number of history list entries of that are saved at logout.

shell contains the name of the file in which the shell resides.

status contains the status returned by the last command.

term specifies the type of terminal in use and corresponds to the environment variable TERM.

time controls automatic timing of commands.

user specifies the current user name and corresponds to the environment variable USER.

verbose prints the words of each command after history substitution.

`Argv`, `cwd`, `home`, `path`, `prompt`, `shell` and `status` are predefined by the shell. `Cwd` and `status` are updated continuously; the others are set only at initialization.

EXAMPLE

```
% set foo=bar
% echo $foo
bar
% set foo=(one two three)
% echo $foo
one two three
% set foo[2]=four
% echo $foo
one four three
```

NOTES

- In all cases, the value is command- and filename-expanded.
- The at-sign (@) command is similiar to *set* but works only with numeric variables.
- The Bourne shell **set** command sets options rather than variables (see *Concentrix System User's Guide).*

SEE ALSO

at_sign, setenv, unset
Concentrix System User's Guide: Using the C Shell, Programming the C Shell, C Shell Reference

set

set shell options (Bourne shell)

The set command sets options and assigns arguments (positional parameters) to $1, $2, etc., in order. If no arguments are given, set prints the values of all names.

---

set [eknptuvx [*arg*...]]

ARGUMENTS

-e If non-interactive, exit immediately if a command fails.

-k Place all keyword arguments in the environment for a command, not just those preceding the command name.

-n Read commands, but do not execute them.

-t Exit after reading and executing one command.

-u Treat unset variables as an error when substituting.

-x Print commands and their arguments as they are executed.

- Turn off the -x and -v options.

arg is the name of the variable to set. With no arguments, *set* displays the values of all names.

DESCRIPTION

These flags can also be used upon invocation of the shell. The current set of flags can be found in $-.

EXAMPLE

```
$ set foo fo fum
$ echo $1
foo
$ echo $2
fo
$ echo $3
fum
```

SEE ALSO

sh
Concentrix System User's Guide: Using the Bourne Shell, Programming the Bourne Shell, Bourne Shell Reference

setenv

set environment variable (C shell)

The setenv command declares and assigns a value to an environment variable and places that variable in the calling environment of all child processes.

---

setenv

setenv *name*

setenv *name value*

ARGUMENTS

name specifies the name of an environment variable.

value is the value to assign.

DESCRIPTION

The first form of the syntax lists all current environment variables. The second form sets the environment variable *name* to the empty string. The third form sets *name* to *value*.

ENVIRONMENT VARIABLES

Environment variables are maintained independently from other shell variables. They are automatically passed by the C shell to each subshell and are available to user programs via the system call execven . By convention, environment variables are upper case and environment strings have the form *name=value*. The following are the default environment variables:

EXINIT is a list of startup commands read by ex, edit, and vi.

HOME is your login directory, set by login from the password file.

PATH is the sequence of directory prefixes that sh, time, nice, etc., apply in searching for a file known by an incomplete path name. The prefixes are separated by colon. The login program sets the following value:

```
PATH=:/usr/ucb:/bin:/usr/bin
```

PRINTER is the name of the default printer to be used by lpr, lpq, and lprm.

SHELL is the file name of the your login shell.

TERM is the kind of terminal for which output is to be prepared. This is used by commands, such as nroff or plot, that can exploit special terminal capabilities. See termcap (*Concentrix System Reference Manual: File Formats)* for details.

TERMCAP describes the terminal in TERM or the name of the termcap file.

USER is your login name.

The environment variables, USER, TERM, and PATH have a special correspondence to the shell variables `user`, `term`, and `path`. Changing one of them automatically changes its corresponding shell variable. Similarly, changing of the the shell variables automatically changes the corresponding environment variable.

SEE ALSO

set, printenv, unsetenv
Concentrix System User's Guide: C Shell Reference

sh

invoke Bourne shell

The sh command invokes the Bourne shell, a program that executes commands read from a terminal or a file.

---

sh [-ceiknrstuvx] [*arg...*]

ARGUMENTS

- if the first character of argument zero is a hyphen, commands are read from $HOME/.profile, if such a file exists. Also turns off the -x and -v options.

-c *string* specifies a command line to be interpreted in the context of the newly created shell.

-e if non-interactive, exits immediately if a command fails.

-i (or if the shell input and output are attached to a terminal (as told by gtty)) specifies this shell is *interactive*. In this case the terminate signal SIGTERM (see sigvec *(System Calls)*) is ignored (so that kill 0 does not kill an interactive shell) and the interrupt signal SIGINT is caught and ignored (so that wait is interruptible). In all cases SIGQUIT is ignored by the shell.

-k places all keyword arguments in the environment for a command, not just those that precede the command name.

-n reads commands, but does not execute them.

-s (or if no arguments remain) specifies that commands are to be read from the standard input. Shell output is written to file descriptor 2.

-t exits after reading and executing one command.

-u treats unset variables as an error when substituting.

-v prints shell input lines as they are read.

-x prints commands and their arguments as they are executed.

COMMANDS

A **simple command** is a sequence of non-blank *words* separated by blanks (a tab or a space). The first word specifies the *name* of the command to be executed. A name is a sequence of letters, digits, or underscores beginning with a letter or underscore. Except as specified below, the remaining words are passed as arguments to the invoked command. The command name is passed as argument 0 (see the system call execve). The

value of a command is its exit status if it terminates normally, or (octal) 200+*status* if it terminates abnormally (see the system call sigvec for a list of status values).

A **pipeline** is a sequence of one or more commands separated by | (or, for historical compatibility, by ^). The standard output of each command but the last is connected by a pipe to the standard input of the next command. Each command is run as a separate process; the shell normally waits for the last command to terminate before prompting for or accepting the next input line. The exit status of a pipeline is the exit status of its last command.

A **list** is a sequence of one or more simple commands or pipelines, separated by ;, &, &&, or ||, and optionally terminated by ; or &. Of these four symbols, ; and & have equal precedence, which is lower than that of && and ||. The symbols && and || also have equal precedence. A semicolon (;) causes sequential execution of the preceding pipeline; an ampersand (&) causes asynchronous execution of the preceding pipeline (the shell does not wait for that pipeline to finish). The symbol && causes the list following to be executed only if the preceding pipeline (or command) returns a zero exit status. With ||, list is executed only if the preceding pipeline (or command) returns a nonzero exit status. NEWLINEs may appear in a list, instead of semicolons, to delimit commands.

A **command** is either a simple command or one of the following constructions. Unless otherwise stated, the value returned by a command is that of the last simple command executed in the construction.

```
for name [ in word ... ] do list done
```

for — Each time a for command is executed, *name* is set to the next word taken from the *in word* list. If *in word ...* is omitted, then the for command executes the *do list* once for each positional parameter that is set (see Parameter Substitution below). Execution ends when there are no more words in the list.

```
case word in [pattern[ | pattern] ... ) list ;; ] ... esac
```

case — A case command executes the *list* associated with the first pattern that matches *word*. The form of the patterns is the same as that used for file-name generation except that a slash, a leading dot, or a dot immediately following a slash need not be matched explicitly.

```
if list then list [ elif list then list ] ... [ else list ] fi
```

if — The *list* following is executed and, if it returns a zero exit status, the *list* following the first then is executed. Otherwise, the *list* following *elif* is executed and, if its value is zero, the *list* following the next then is executed. Failing that, the *else list* is executed. If no *else list* or *then list* is executed, then the if command returns a zero exit status.

```
while list do list done
```

while — A while command repeatedly executes the *while list* and, if the exit status of the last command in the list is zero, executes the *do list;* otherwise the loop terminates. If no commands in the *do list* are executed, then the while command returns a zero exit status; until may be used in place of while to negate the loop termination test.

(list) Execute list in a subshell.

{list;} list is simply executed.

name () *{list;}*
Define a function which is referenced by *name*. The body of the function is the list of commands between { and }. Execution of functions is described below.

The following words are only recognized as the first word of a command and when not quoted:

```
if then else elif fi case esac for while until do done { }
```

Comments A word beginning with # causes that word and all the following characters up to a NEWLINE to be ignored.

COMMAND SUBSTITUTION

The standard output from a command enclosed in a pair of back quotes (``) may be used as part or all of a word; trailing NEWLINEs are removed.

PARAMETER SUBSTITUTION

A **parameter** is a name, a digit, or any of the characters *, @, #, ?, –, $, and ! .

The character $ is used to introduce substitutable parameters. Positional parameters (digits) can be assigned values by the set command. Keyword parameters (variables) can be assigned values by writing:

```
name=value [ name=value ] ...
```

${parameter}
The value, if any, of *parameter* is substituted. The braces are required only when *parameter* is followed by a letter, digit, or underscore that is not to be interpreted as part of its name. If *parameter* is * or @, all the positional parameters, starting with $1, are substituted (separated by spaces). $0 is set from argument zero when the shell is invoked.

If the colon (:) is omitted from the following expressions, the shell only checks whether the parameter is set or not.

${parameter:–word}
If *parameter* is set and is nonnull, substitute its value; otherwise substitute word.

${parameter:=word}
If *parameter* is not set or is null set it to word; the value of *parameter* is substituted. Positional parameters may not be assigned to in this way.

${parameter:?word}
If *parameter* is set and is nonnull, substitute its value; otherwise, print *word* and exit from the shell. If *word* is omitted, a standard message is printed.

${parameter:+word}
If *parameter* is set and is nonnull, substitute *word*; otherwise substitute nothing.

In the above, *word* is not evaluated unless it is to be used as the substituted string. (So that, for example, echo ${d:-`pwd`}*pwd* is executed only if *d* is not set or is null.)

The following parameters are automatically set by the shell:

\# The number of positional parameters in decimal.

\- Flags supplied to the shell on invocation or by the set command.

? The decimal value returned by the last synchronously executed command.

$ The process number of this shell.

! The process number of the last background command invoked.

The following parameters are used but not set by the shell:

HOME The default argument (home directory) for the cd command.

PATH The search path for commands.

CDPATH
The search path for the cd command.

MAIL If this parameter is set to the name of a mail file and the MAILPATH parameter is not set, the shell informs the user of the arrival of mail in the specified file.

MAILPATH
A colon (:) separated list of filenames. If this parameter is set, the shell informs the user of the arrival of mail in any of the specified files. Each filename can be followed by % and a message that will be printed when the modification time changes. The default message is :"you have mail."

PS1 Primary prompt string, by default "$ ".

PS2 Secondary prompt string, by default "> ".

SHELL
When the shell is invoked, it scans the environment for this name. If it is found and there is an 'r' in the filename part of its value, the shell becomes a restricted shell.

BLANK INTERPRETATION

After parameter and command substitution, any results of substitution are scanned for internal field separator characters (those found in IFS) and split into distinct arguments where such characters are found. Explicit null arguments (" or ") are retained. Implicit null arguments (those resulting from parameters that have no values) are removed.

FILE NAME GENERATION

Following substitution, each command word is scanned for the characters *, ?, and [. If one of these characters appears the word is regarded as a pattern. The word is replaced with alphabetically sorted filenames that match the pattern. If no filename is found that matches the pattern, the word is left unchanged. The character . at the start of a filename or immediately following a /, as well as the character / itself, must be matched explicitly.

* Matches any string, including the null string.

? Matches any single character.

[...] Matches any one of the enclosed characters. A pair of characters separated by – matches any character lexically between the pair, inclusive. If the first character following the opening "[" is a "!" any character not enclosed is matched.

QUOTING

The following characters have a special meaning to the shell and cause termination of a word unless quoted:

; & ( ) | ^ < > NEWLINE* SPACES* TABS*

A character may be quoted (made to stand for itself) by preceding it with a \. The pair \NEWLINE is ignored. All characters enclosed between a pair of single quote marks (''), except a single quote, are quoted. Inside double quote marks (""), parameter and command substitution occurs and \ quotes the characters \, `, ", and $. "$*" is equivalent to "$1 $2 ...", whereas "$@" is equivalent to "$1" "$2" ....

PROMPTING

When used interactively, the shell prompts with the value of PS1 before reading a command. If at any time a NEWLINE is typed and further input is needed to complete a command, the secondary prompt (PS2) is issued.

INPUT/OUTPUT

Before a command is executed, its input and output may be redirected using a special notation interpreted by the shell. The following may appear anywhere in a simple command or may precede or follow a command and are not passed on to the invoked command; substitution occurs before *word* or *digit* is used:

<word Use file word as standard input (file descriptor 0).

>word Use file word as standard output (file descriptor 1). If the file does not exist it is created; otherwise, it is truncated to zero length.

>>word Use file word as standard output. If the file exists output is appended to it; otherwise, the file is created.

<<[–]word The shell input is read up to a line that is the same as *word*, or to an end–of–file. The resulting document becomes the standard input. If any

character of *word* is quoted, no interpretation is placed upon the characters of the document; otherwise, parameter and command substitution occurs, (unescaped) \NEWLINE is ignored, and \ must be used to quote the characters \, $, `, and the first character of *word*. If - is appended to <<, all leading tabs are stripped from *word*, and from the document.

<&digit Use the file associated with file descriptor *digit* as standard input. Similarly for the standard output using >&*digit*.

<&- The standard input is closed. Similarly for the standard output using >&-.

If any of the above is preceded by a digit, the file descriptor associated with the file is that specified by the digit (instead of the default 0 or 1). For example:

```
... 2>&1
```

associates file descriptor 2 with the file currently associated with file descriptor 1.

If a command is followed by &, the default standard input for the command is the empty file (/dev/null). Otherwise, the environment for the execution of a command contains the file description of the invoking shell as modified by input/output specification.

ENVIRONMENT

The environment (see the system call execve) is a list of name-value pairs passed to an executed program in the same way as a normal argument list. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a parameter for each name found, giving it the corresponding value. If the user modifies the value of any of these parameters or creates new parameters, none of these affects the environment unless the export command is used to bind the shell's parameter to the environment (see also set -a). A parameter may be removed from the environment with the unset command. The environment seen by any executed command is thus composed of any unmodified name-value pairs originally inherited by the shell, minus any pairs removed by unset, plus any modifications or additions, all of which must be noted in export commands.

The environment for any simple command may be augmented by prefixing it with one or more assignments to parameters. Thus these two lines are equivalent:

```
TERM=450 cmd args
(export TERM; TERM=450; cmd args)
```

If the -k flag is set, all keyword arguments are placed in the environment, even if they occur after the command name. The following first prints a=b c and c:

```
echo a=b c
set -k
echo a=b c
```

SIGNALS

The INTERRUPT and QUIT signals for an invoked command are ignored if the command is followed by &; otherwise signals have the values inherited by the shell from its parent, with the exception of signal 11 (but see also the trap command).

EXECUTION

Each time a command is executed, the above substitutions are carried out. If the command name matches one of the Special Commands listed below, it is executed in the shell process. If the command name does not match a Special Command, but matches the name of a defined function, the function is executed in the shell process (note how this differs from the execution of shell procedures). The positional parameters $1, $2, ... are set to the arguments of the function. If the command name matches neither a Special Command nor the name of a defined function, a new process is created and an attempt is made to execute the command via the system call exec.

DIAGNOSTICS

- Errors detected by the shell, such as syntax errors cause the shell to return a non zero exit status. If the shell is being used non interactively, execution of the shell file is abandoned. Otherwise, the shell returns the exit status of the last command executed (see exit).

NOTES

- If double left angle bracket (<<) is used to provide standard input to an asynchronous process invoked by ampersand (&), the shell gets mixed up about naming the input document. A garbage file /tmp/sh* is created, and the shell complains about not being able to find the file by another name.

- A *cmdfile* which has execute permission but is not a binary executable and is not a shell script directly executable by Concentrix-2800 is further examined by the shell which is attempting to execute the command. If the first character is a #(pound sign) then *cmdfile* is deemed to be a file of csh commands. If the first character is NOT a #, then *cmdfile* is deemed to be a file of sh commands. If necessary, one shell will exec the other to handle the commands.

  The csh somewhat complicates this by using the string associated with the alias "shell" as the interpreter to use. Failing that, it examines the first character and uses /bin/sh if it is not #. Otherwise (if it is #), csh examines the shell variable "shell" and uses the interpreter associated with it. Finally, failing that, it falls back to using /bin/csh.

  Both shells exec the interpreter which is finally chosen with *cmdfile* as the argument.

FILES

$HOME/.profile
/tmp/sh*
/dev/null

SPECIAL COMMANDS

The Bourne Shell Special Commands are listed here. An individual man page is supplied for each one in this manual. These commands are executed in the shell process, and except where specified no input output redirection is permitted.

:	No effect
. file	read and execute commands from *file* and return. Note: There must be a space between "." and the filename.)
break	exit from the enclosing for or while loop
continue	resume next iteration of the enclosing for or while loop
cd [*arg*]	change current directory to *arg*
eval	read *arg* as input and execute resulting command(s)
exec	execute command specified without creating new process
exit	causes noninteractive shell to exit
export	marks names for automatic export to *environment* of subsequently executed commands
login	execute login *arg*
read	reads a line from standard input; assigns successive words to variables *name* in order
readonly	mark given names readonly
set	set shell environment variables
shift	rename positional parameters
times	print accumulated user and system times
trap	reads *arg* when signal received
umask	set user file creation mask to octal value
wait	wait for process to terminate and report termination status

SEE ALSO

csh, set, test
Concentrix-2800 System Reference Manual: System Calls and C Library Routines: execven, execvec
Concentrix System User's Guide: Using the Bourne Shell and *Bourne Shell Reference*

shift

shift members of variable (C shell)

The shift command shifts the elements of the specified variable to the left, discarding the first element.

---

shift [*variable*]

ARGUMENTS

variable must have at least two words of value; if omitted, the default is `argv`.

SEE ALSO

set

shift

rename positional parameters (Bourne shell)

The shift command renames the positional parameters (starting at $2) from $*n* to $*n*–1.

---

shift

SEE ALSO

set

size

display size of an object file

The size command displays the (decimal) number of bytes required by the text, data, and bss portions, and their sum in hex and decimal, of each object-file argument.

---

size [*object...*]

ARGUMENTS

object specifies an object file (default is a.out).

SEE ALSO

Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: a.out

sleep

suspend execution for an interval

The sleep command suspends execution for a specified period of time.

---

sleep *time*

ARGUMENTS

time specifes the time to sleep in seconds.

DESCRIPTION

Sleep is used to execute a command after a certain amount of time as in:

```
(sleep 105; command)&
```

or to execute a command every so often, as in:

```
while true
do
        command
        sleep 37
done
```

NOTES

- *Time* must be less than 2,147,483,647 seconds.

SEE ALSO

Concentrix-2800 System Reference Manual: System Calls and C Library Routines: alarm, sleep, setitimer

soelim

eliminate .so's from nroff input

The soelim command reads the specified nroff source files and performs the textual inclusion implied by .so directives. It is useful because programs such as tbl do not process .so directives, which would otherwise prevent the placement of individual tables in separate files to be run as a part of a large document.

---

soelim [*file...*]

ARGUMENTS

file specifies an input file. A hyphen specifies the standard input.

DESCRIPTION

The directives must be of the following form and must appear at the beginning of input lines:

```
.so somefile
```

Inclusion can be suppressed by using a single quote instead of a period, i.e.

```
'so /usr/lib/tmac.s
```

A sample usage of soelim would be

```
soelim exum?.n | tbl | nroff -ms | col | lpr
```

NOTES

- **The format of the source commands must be clean: exactly one blank must precede and no blanks follow the file name.**

SEE ALSO

colcrt, more

sort

sort or merge files

The sort command sorts lines of all the named files together and writes the result on the standard output.

---

sort [-bdfinrmcut*x*] [+*pos1* [-*pos2*]]... [-o *name*] [-T *directory*] [*name...*]

ARGUMENTS

Ordering is affected globally by the following options:

-b ignores leading blanks (spaces and tabs) in field comparisons.

-d specifies "dictionary" order: only letters, digits and blanks are significant in comparisons.

-f folds upper case letters onto lower case.

-i ignores characters outside the ASCII range 040–0176 (octal) in nonnumeric comparisons.

-n sorts by arithmetic value, an initial numeric string, consisting of optional blanks, optional minus sign, and zero or more digits with optional decimal point. Option n implies option b.

-r reverses the sense of comparisons.

-t*x* specifies the character *x* to be used as a field separator.

+*pos1*
-*pos2* restricts a sort key to a field beginning at *pos1* and ending just before *pos2*, each of which has the form:

m.n[bdfinr]

m tells a number of fields to skip from the beginning of the line.

n tells a number of characters to skip further.

If any flags are present, they override all the global ordering options for this key. If the b option is in effect *n* is counted from the first nonblank in the field; b is attached independently to *pos2*. A missing .*n* means .0; a missing -*pos2* means the end of the line. Under the -t*x* option, fields are strings separated by *x*; otherwise fields are nonempty nonblank strings separated by blanks.

These option arguments are also understood:

-m merges but does not sort (the input files are already sorted).

-c checks that the input file is sorted according to the ordering rules; give no output unless the file is out of sort.

-o *name*
: specifies the name of an output file, which can be the same name as one of the input files.

-T *directory*
: specifies the name of a directory in which to store temporary files.

-u
: suppresses all but one in each set of equal lines. Ignored bytes and bytes outside keys do not participate in this comparison.

name
: specifies the name of an input file. A hyphen specifies the standard input, which is the default.

DESCRIPTION

The default sort key is an entire line. Default ordering is lexicographic by bytes in machine collating sequence.

When there are multiple sort keys, later keys are compared only after all earlier keys compare equal. Lines that otherwise compare equal are ordered with all bytes significant.

EXAMPLE

Display in alphabetical order all the unique spellings in a list of words. Capitalized words differ from uncapitalized.

```
sort -u +0f +0 list
```

Display the password file passwd sorted by user id number (the 3rd colon-separated field). See *Concentrix System Reference: Miscellaneous Routines, Special Files, and File Formats* for more information about passwd.

```
sort -t: +2n /etc/passwd
```

Display the first instance of each month in an already sorted file of (month day) entries. The options -um with just one input file make the choice of a unique representative from a set of equal lines predictable.

```
sort -um +0 -1 dates
```

DIAGNOSTICS

- Comments and exits with nonzero status for various trouble conditions and for disorder discovered under option -c.

NOTES

- Very long lines are silently truncated.

FILES

/usr/tmp/stm*, /tmp/* first and second tries for temporary files

SEE ALSO

uniq, comm, rev, join

sortbib

sort bibliographic database (unsupported)

The sortbib command sorts files of records containing refer key-letters by user-specified keys. Sortbib reads through each *database* and pulls out key fields, which are sorted separately. The sorted key fields contain the file pointer, byte offset, and length of corresponding records. These records are delivered using disk seeks and reads, so *sortbib* cannot be used in a pipeline to read standard input.

---

sortbib [**-s***KEYS*] *database...*

ARGUMENTS

-s*KEYS* specifies keys on which to sort. For example, -sATD sorts by author, title, and date, while -sA+D sorts by all authors, and date. Sort keys past the fourth are not meaningful.

database specifies the name of a database file. No more than 16 databases can be sorted together at one time.

DESCRIPTION

Records can be separated by blank lines, or by `.[` and `.]` delimiters, but the two styles can not be mixed together. Records longer than 4096 characters are truncated.

By default, sortbib alphabetizes by the first %A and the %D fields, which contain the senior author and date.

Sortbib sorts on the last word on the %A line, which is assumed to be the author's last name. A word in the final position, such as `jr.` or `ed.`, is ignored if the previous name ends with a comma. Authors with two-word last names or unusual constructions can be sorted correctly by using the nroff convention `\0` in place of a blank. A %Q field is considered to be the same as %A, except sorting begins with the first, not the last, word.

Sortbib sorts on the last word of the %D line, usually the year. It also ignores leading articles (like A or The) when sorting by titles in the %T or %J fields; it ignores articles of any modern European language. If a sort-significant field is absent from a record, sortbib places that record before other records containing that field.

NOTES

- Records with missing author fields should probably be sorted by title.

SEE ALSO

refer, addbib, roffbib, indxbib, lookbib

source

execute commands from file (C shell)

The source command reads and executes commands from a file.

---

source [-h] ***name***

ARGUMENTS

-h places the commands in the history list without being executed.

name is the name of a file containing commands to execute.

NOTES

- The file does not require execute access mode.
- Source commands can be nested. If nested too deeply, the shell may run out of file descriptors.
- An error in a command file at any level terminates all nested source commands.

spell

find spelling errors

The spell command collects words from the named documents, and looks them up in a spelling list.

---

spell [-v] [-b] [-x] [-d *hlist*] [-s *hstop*] [-h *spellhist*] [*file...*]

ARGUMENTS

-v	displays all words not literally in the spelling list, and indicates plausible derivations from spelling list words.
-b	checks British spelling. Besides preferring `centre`, `colour`, `speciality`, `travelled`, etc., this option insists upon `-ise` in words like `standardise`, Fowler and the OED to the contrary notwithstanding.
-x	displays every plausible stem with an equal sign for each word.
-d *hlist*	specifies a spelling list.
-s *hstop*	specifies a stop list to filter out misspellings (e.g. thier=thy-y+ier) that would otherwise pass.
-h *spellhist*	specifies a history in which copies of all output can be accumulated.
file	specifies an input file. The default is the standard input.

The default spelling list, stop list, and history list are shown in the Files section.

DESCRIPTION

Words that neither occur among nor are derivable (by applying certain inflections, prefixes or suffixes) from words in the spelling list are displayed on the standard output. Spell ignores most troff, tbl and eqn constructions.

The spelling list is based on many sources. While it is more haphazard than an ordinary dictionary, it is also more effective with proper names and popular technical words. Coverage of the specialized vocabularies of biology, medicine and chemistry is light.

Two routines help maintain the hash lists used by spell. Spellin combines words to create a new hashed list. Spellout looks up words and displays those that are missing from or (optionally) present on the hashed *list* file.

NOTES

- The spelling list's coverage is uneven; new installations will probably wish to monitor the output for several months to gather local additions.
- British spelling was done by an American.

FILES

/usr/dict/hlist[ab]	hashed spelling lists, American & British, default for -d
/usr/dict/hstop	hashed stop list, default for -s
/dev/null	history file, default for -h
/tmp/spell.$$*	temporary files
/usr/lib/spell	

SEE ALSO

spellin, spellout, deroff, sort, tee, sed

spellin

create new hash list for spell

The spellin command combines a set of words, one per line, from the standard input and the preexisting list file and places a new list on the standard output.

---

spellin [*list*]

ARGUMENTS

list specifies the name of a list file. If omitted, a new list is created from scratch.

EXAMPLE

To verify that the word hookey is not on the default spelling list, add it to your own private list, and then use it with spell:

```
echo hookey | spellout /usr/dict/hlista
echo hookey | spellin /usr/dict/hlista > myhlist
spell -d myhlist huckfinn
```

SEE ALSO

spell, spellout

spellout

report on hash list for spell

The spellout command looks up each of a set of words, one per line, from the standard input and displays on the standard output those that are missing from the hashed *list* file.

---

spellout [-d] *list*

ARGUMENTS

-d inverts the operation, that is, displays files that are present on (rather than missing from) the hashed *list* file.

list specifies the name of a list file.

EXAMPLE

To verify that the word hookey is not on the default spelling list, add it to your own private list, and then use it with spell:

```
echo hookey | spellout /usr/dict/hlista
echo hookey | spellin /usr/dict/hlista > myhlist
spell -d myhlist huckfinn
```

SEE ALSO

spell, spellin

spline

interpolate smooth curve

The spline command takes pairs of numbers from the standard input as abcissas and ordinates of a function. It produces a similar set, which is approximately equally spaced and includes the input set, on the standard output. The cubic spline output has two continuous derivatives, and sufficiently many points to look smooth when plotted, for example by graph.

---

spline [*option*...]

ARGUMENTS

The following options are recognized, each as a separate argument.

-a [*s*] supplies abscissas automatically (they are missing from the input); *s* specifies the spacing (default is one).

-k *n* specifies the value of the constant *k* used in the boundary value computation:

$y(0)'' = ky(1)''$, $y(n)'' = ky(n-1)''$

By default $k = 0$.

-n spaces output points so that approximately *n* intervals occur between the lower and upper *x* limits. By default $n = 100$.

-p makes output periodic, that is, matches derivatives at ends. The first and last input values should normally agree.

-x *xmin* [*xmax*]
specifies lower (and upper) *x* limits. Normally these limits are calculated from the data. Automatic abcissas start at the lower limit (default 0).

DIAGNOSTICS

- When data is not strictly monotone in *x*, *spline* reproduces the input without interpolating extra points.

NOTES

- A limit of 1000 input points is enforced silently.

SEE ALSO

graph, plot
R. W. Hamming, *Numerical Methods for Scientists and Engineers*, 2nd ed., 349ff

split

split a file into pieces

The split command reads a file and splits it onto a set of output files.

---

split [-n] [*file* [*name*]]

ARGUMENTS

-n specifies the number of lines to write onto each output file (default 1000).

file specifies an input file. A hyphen specifies the standard input file (the default).

name specifies the name of the output files. The first output file has the specified name with aa appended, and so on lexicographically. The default is x.

stop

stop background job (C shell)

The stop command stops a job that is executing in the background. The job can be restarted with the bg or fg command.

---

stop [*%job*]...

ARGUMENTS

%job specifies a job number. The default is the current background job.

SEE ALSO

bg, jobs

strings

find printable strings in a binary file

The strings command looks for ASCII strings in a binary file. A string is any sequence of four or more printing characters ending with a newline or a null. Strings is useful for identifying random object files and many other things.

---

strings [-] [-o] [-*number*] *file...*

ARGUMENTS

-	expands the search. By default, *strings* looks only in the initialized data space of object files.
-o	precedes each string by its offset in the file (in octal).
-*number*	specifies a minimum string length (default is four).
file	specifies a file to search for strings.

NOTES

- The algorithm for identifying strings is extremely primitive

SEE ALSO

od

strip

remove symbols and relocation bits

The strip command removes the symbol table and relocation bits ordinarily attached to the output of the assembler and loader. This is useful to save space after a program has been debugged.

---

strip *name...*

DESCRIPTION

The effect of strip is the same as use of the **-s** option of ld.

FILES

/tmp/stm?	temporary file

SEE ALSO

ld

stty

set terminal options

The stty command sets certain I/O options on the current output terminal, placing its output on the diagnostic output.

---

stty [*option...*]

ARGUMENTS

With no argument, stty reports the speed of the terminal and the settings of the options that are different from their defaults.

0 hangs up phone line immediately.

50 75 110 134 150 200 300 600 1200 1800 2400 4800 9600 extb
sets terminal baud rate to the number given, if possible.

all displays all normally used option settings.

everything displays everything stty knows about.

bs0 bs1 selects style of delay for backspace.

cbreak makes each character available to read *(System Calls)* as received; no erase and kill processing, but all other processing (interrupt, suspend, ...) is performed.

–cbreak makes characters available to *read* only when newline is received.

cooked same as –raw.

cr0 cr1 cr2 cr3
selects style of delay for carriage return (see ioctl *(System Calls)*).

dec sets all modes suitable for Digital Equipment Corp. operating systems users; (erase, kill, and interrupt characters to ^?, ^U, and ^C, decctlq and "newcrt".)

echo echos back every character typed.

–echo disables echo.

ek sets erase and kill characters to sharp sign and at sign.

even allows even parity input.

–even disallows even parity input.

ff0 ff1 selects style of delay for form feed.

lcase maps upper case to lower case.

-lcase disables case mapping.

nl0 nl1 nl2 nl3
selects style of delay for linefeed.

odd allows odd parity input.

-odd disallows odd parity input.

-nl allows carriage return for new-line, and output CR-LF for carriage return or new-line.

nl accepts only new-line to end lines.

raw sets raw mode input (no input processing (erase, kill, interrupt, ...); parity bit is passed back).

-raw negates raw mode.

size prints the terminal (window) sizes on the standard output: rows followed by columns.

speed prints the terminal speed on the standard output.

tab0 tab1 tab2 tab3
selects style of delay for tab.

-tabs replaces tabs by spaces when printing.

tabs preserves tabs.

tandem enables flow control, so that the system sends out the stop character when its internal queue is in danger of overflowing on input, and sends the start character when it is ready to accept further input.

-tandem disables flow control.

tek sets all modes suitable for Tektronix 4014 terminal.

ti700 sets all modes suitable for Texas Instruments 700 series terminal.

tn300 sets all modes suitable for a General Electric TermiNet 300.

tty33 sets all modes suitable for the Teletype Corporation Model 33 terminal.

tty37 sets all modes suitable for the Teletype Corporation Model 37 terminal.

vt05 sets all modes suitable for Digital Equipment Corp. VT05 terminal.

For the following commands, that take a character argument *c*, you can also specify *c* as the `u` or `undef`, to set the value to be undefined. A value of `^x`, a two-character sequence, is also interpreted as a control character, with `^?` representing DELETE .

brk *c* sets break character to *c* (default undefined.) This character is an additional character causing wakeup.

eof *c* sets end of file character to *c* (default ^D.)

erase *c* sets erase character to *c* (default is sharp sign (#), but often reset to ^H.)

intr *c* sets interrupt character to *c* (default DELETE or ^?, but often reset to ^C.)

kill *c* sets kill character to *c* (default is at sign (@), but often reset to ^U.)

quit *c* sets quit character to *c* (default ^\.)

start *c* sets start character to *c* (default ^Q.)

stop *c* sets stop character to *c* (default ^S.)

The following commands take a numeric argument *n*.

cols *n* synonym for *columns*.

columns *n* records the terminal size as having *n* columns.

rows *n* records the terminal size as having *n* rows.

A teletype driver that supports the job control processing of *csh* and more functionality than the basic driver is fully described in tty*(Special Files)*. The following options apply only to it.

ctlecho echoes control characters as ^x (and delete as ^?.) Print two backspaces following the EOT character (CTRL D).

-ctlecho echoes control characters as themselves; in cooked mode EOT (CTRL D) is not echoed.

crt sets options for a video terminal (crtbs, ctlecho and, if >= 1200 baud, crterase and crtkill.)

crtbs echos backspaces on erase characters.

crterase wipes out erased characters with backspace-space-backspace.

-crterase leaves erased characters visible; just backspace.

crtkill wipes out input on like kill ala crterase.

-crtkill echoes just line kill character and a newline on line kill.

decctlq after output is suspended (normally by CTRL S), only a start character (normally CTRL Q) restarts it. This is compatible with DEC's vendor supplied systems.

-decctlq after output is suspended, any character typed restarts it; the start character restarts output without providing any input. (This is the default.)

etxack Diablo style etx/ack handshaking (not implemented).

flusho output is being discarded usually because user pressed CTRL O (internal state bit).

-flusho output is not being discarded.

litout sends output characters without any processing.

-litout does normal output processing, inserting delays, etc.

mdmbuf starst/stops output on carrier transitions (not implemented).

-mdmbuf returns error if write attempted after carrier drops.

new uses new driver (switching flushes typeahead).

nohang disables sending hangup signal if carrier drops.

-nohang sends hangup signal to control process group when carrier drops.

pass8 passes all 8 bits through on input, in any mode.

-pass8 stips the 0200 bit on input, except in raw mode.

pendin input is pending after a switch from cbreak to cooked and is reinput when a read becomes pending or more input arrives (internal state bit).

-pendin input is not pending.

tilde converts tilde character to a grave accent on output (for Hazeltine terminals).

-tilde disables tilde conversion.

tostop background jobs stop if they attempt terminal output.

-tostop allows output from background jobs to the terminal.

The following special characters are applicable only to the new teletype driver and are not normally changed.

dsusp *c* sets delayed suspend process character to *c* (default CTRL Y).

flush *c* sets flush output character to *c* (default CTRL O).

rprnt *c* sets reprint line character to *c* (default CTRL R).

susp *c* sets suspend process character to *c* (default CTRL Z).

werase *c* sets word erase character to *c* (default CTRL W). lnext *c* set literal next character to *c* (default CTRL V).

SEE ALSO

Concentrix-2800 System Reference Manual: System Calls and C Library Routines: ioctl
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: tabs, tset, tty

style

analyze document surface characteristics (unsupported)

The style command analyzes the surface characteristics of the writing style of a document. It reports on readability, sentence length and structure, word length and usage, verb type, and sentence openers.

---

style [-ml] [-mm] [-a] [-e] [-l *num*] [-r *num*] [-p] [-P] *file...*

ARGUMENTS

-ml skips lists (should be used if the document contains many lists of non-sentences).

-mm overrides the default macro package -ms.

The other options locate sentences with certain characteristics.

-a displays all sentences with their length and readability index.

-e displays all sentences that begin with an expletive.

-p displays all sentences that contain a passive verb.

-l*num* displays all sentences longer than *num*.

-r*num* displays all sentences whose readability index is greater than *num*.

-P displays parts of speech of the words in the document.

NOTES

- Because style runs deroff before looking at the text, formatting header files should be included as part of the input.
- Use of non-standard formatting macros may cause incorrect sentence breaks.

SEE ALSO

deroff, diction

su

substitute userid (superuser)

The su command invokes an interactive shell with a specific user identification. The default userid is root, which has super-user privilege.

---

su [-f] [-] [*userid*]

ARGUMENTS

userid specifies a user identification. The default is root, the superuser.

-f prevents *su* from executing the .cshrc file, thus making *su* start up faster.

- simulates a full login.

DESCRIPTION

Su requests the password of the specified *userid* and, if the password and other privileges are correct, invokes the desired shell.

The user environment, including the current directory, is unchanged except for HOME and SHELL, which are taken from the password file for the user being substituted. The new user ID stays in force until the Shell exits.

NOTES

- Only users in group 0 can use *su* to become root, even if they supply the correct password.
- To remind a superuser of his or her responsibilities, the shell substitutes the sharp sign (#) for its usual prompt.

SEE ALSO

sh, csh

sum

sum and count blocks in a file

The sum command calculates and displays a 16-bit checksum for the named file, and also displays the number of blocks in the file. It is typically used to look for bad spots, or to validate a file communicated over some transmission line.

---

sum *file...*

DIAGNOSTICS

- `Read error` is indistinguishable from end of file on most devices; check the block count.

SEE ALSO

wc

suspend

stop current shell (C shell)

The suspend command stops the current shell, as if it had been sent a stop signal with CTRL Z.

---

suspend

SEE ALSO

bg, fg, stop

switch

multi-way branch control structure (C shell)

The switch command is similar to the case structure in C, a dispatch-table mechanism. It is intended for use in shell scripts.

---

switch (word)
case str1:
    command
        .
        .
        .
    breaksw
.
.
.
case *strn*:
    command
        .
        .
        .
    breaksw
default:
    command
        .
        .
        .
    breaksw
endsw

ARGUMENTS

word is the string on which control is dispatched.

str1...strn is a string that is matched against *word*. The file metacharacters *, ? and [] can be used in the *case* labels, which are variable expanded.

command is any valid command.

DESCRIPTION

The shell searches for the first **case** string that matches the dispatch *word* and executes the commands in its clause. If none match, the shell executes the commands in the

`default` clause, if present. The command *breaksw* causes execution to continue after the *endsw*. Otherwise control can fall through *case* labels and *default* labels as in C. If no label matches and there is no *default*, execution continues after the *endsw*.

NOTES

- Each *case* label and the *default* label must appear at the beginning of a line.
- A very common mistake in shell scripts is to use *break* rather than *breaksw* in switch commands.
- If a case string matches the dispatch *word* but has no command block, control passes to the next command block. Thus, you can have a construct like:

```
case str1:
case str2:
case str3:
        command
        breaksw
```

NOTES

- The Bourne shell case command is similar in function to switch.

SEE ALSO

foreach, if, if-then-else, while

symorder

rearrange name list

The symorder command reorders the name list (symbol table) of an executable file in order to optimize access to particular symbols. Access is generally by debuggers or the C shell for statistics.

---

symorder *orderlist symbolfile*

ARGUMENTS

orderlist specifies a file containing symbols to be found in symbolfile, one symbol per line.

symbolfile specifies a file that is updated in place to put the requested symbols first in the symbol table, in the order specified. This is done by swapping the old symbols in the required spots with the new ones. If all of the order symbols are not found, an error is generated.

DESCRIPTION

This program was specifically designed to cut down on the overhead of getting symbols from `/vmunix`.

SEE ALSO

Concentrix-2800 System Reference Manual: System Calls and C Library Routines: nlist

syscfg

show system configuration

Displays information about the hardware configuration of the system on which it executes.

---

syscfg [-d]

ARGUMENTS

-d prints the device configuration.

DESCRIPTION

The display includes the name of the system, its serial number, the amount of physical memory, the size of the CE cache, the amount of physical memory available to the user, the sizes of the paging and swapping files, the number of IPs, and the number of CEs with the breakdown (attached and detached).

The expanded display (-d option) includes information on the devices attached to each IP.

EXAMPLES

The following example demonstrates a long display for an FX/2800 Series machine:

```
System Configuration for FX2800 buddy (serial # 0 ):
    Memory Size    = 256 Mb
    CE-Cache Size = 1024 Kb
    User Memory    = 224.9 Mb
    Paging File    = 132.3 Mb
    Swapping File = 529.0 Mb
    Number of IPs =  1
    Number of CEs =  8
        Detach/Attach = 5/4
            Cluster 0:  CE00 CE01 CE04 CE05
IP-0:
  xdv0 (port=0) (major= 96) (addr=0xfd00ee40) Xylogics 752/754 Disk Controller
         Drive(s) Online: 0 1
  xdv1 (port=0) (major= 96) (addr=0xfd00ee50) Xylogics 752/754 Disk Controller
         Drive(s) Online: 0 1
  xdv2 (port=0) (major= 96) (addr=0xfd00ee60) Xylogics 752/754 Disk Controller
         Drive(s) Online: 0 1
  xdv3 (port=0) (major= 96) (addr=0xfd00ee70) Xylogics 752/754 Disk Controller
         Drive(s) Online: 0 1
  xdv4 (port=0) (major= 96) (addr=0xfd00ee80) Xylogics 752/754 Disk Controller
         Drive(s) Online: 0 1 2 3
 scsi0 (port=0) (major=165) (addr=0xfd00c000)
  imb0 (port=0) (major=224) (addr=0x0)
   en0 (port=0) (major=206) (addr=0xfe000000)
```

tabs

set terminal tabs

The tabs command sets the tabs on a variety of terminals.

---

tabs [-n] ***terminal***

ARGUMENTS

-n disables left margin indentation.

terminal specifies a terminal type. Various terminal names are recognized; the default is, however, suitable for most 300 baud terminals.

NOTES

- It is recommended that you use tset.

SEE ALSO

stty
Concentrix System User's Guide: Getting Started

tail

display the last part of a file

The tail command copies the named file to the standard output beginning at a designated place.

---

tail [+number [lbc] [fr]] [*file*]

tail [–number [lbc] [fr]] [*file*]

DESCRIPTION

+number begins copying at distance *number* from the beginning of the input.

–number begins copying at distance *number* from the end of the input.

lbc counts *number* in units of lines, blocks or characters, according to the appended option l, b or c. By default, counting is by lines.

f causes tail to not quit at end of file, but rather wait and try to read repeatedly in hopes that the file will grow.

r causes tail to display lines from the end of the file in reverse order. The default for r is to display the entire buffer this way.

file specifies an input file. By default, the standard input is used.

NOTES

- Tails relative to the end of the file are stored in a 4096-byte buffer, and thus are limited in length.
- Various kinds of anomalous behavior can happen with character special files.

SEE ALSO

dd, head

talk

talk to another user

The talk command is a visual communication program that copies lines from your terminal to that of another user.

---

talk *person* [*ttyname*]

ARGUMENTS

person is the person's login name. If you wish to talk to a user on another machine, *person* can take one of the following forms:

host!user
host.user
host:user
user@host

though *user@host* is perhaps preferred.

ttyname indicates the appropriate terminal name when you want to talk to a user who is logged in more than once.

DESCRIPTION

When first called, talk sends the message

```
Message from TalkDaemon@his_machine...
talk: connection requested by your_name@your_machine.
talk: respond with: talk your_name@your_machine
```

to the user you wish to talk to. At this point, the recipient should reply by typing

```
talk your_name@your_machine
```

It doesn't matter from which machine the recipient replies, as long as his login name is the same. Once communication is established, the two parties can type simultaneously, with their output appearing in separate windows. Typing CTRL L causes the screen to be redisplayed, while your erase, kill, and word kill characters work in talk as normal. To exit, just type your interrupt character; talk then moves the cursor to the bottom of the screen and restores the terminal.

Permission to talk can be denied or granted by use of the *mesg* command. At the outset talking is allowed. Certain commands, in particular nroff and pr disallow messages in order to prevent messy output.

NOTES

- The 4.3BSD version of talk uses a protocol that is not compatible with the 4.2BSD version of talk.

FILES

/etc/hosts	to find the recipient's machine
/etc/utmp	to find the recipient's tty

SEE ALSO

mesg, who, mail, write

tar

tape archiver

The tar command saves and restores multiple files on a single file (usually a magnetic tape, but can be any file).

---

tar [*key*] [[**-C**] *name*]...

ARGUMENTS

key is a string of characters consisting of one command letter and optionally one or more command modifiers as described below.

-C causes tar to perform a chdir *(System Calls)* to *name*. This allows multiple directories not related by a close common parent to be archived using short relative path names (see example below).

name is a file or directory name specifying the files to save or restore. In all cases, a directory name refers to the files and (recursively) subdirectories of that directory. If you save an absolute pathname, *tar* restores the same absolute pathname. Thus, to restore files to specific directories, use relative pathnames.

COMMANDS

r writes the named files on the end of the tape. The **c** command letter implies **r**. You cannot append to a cartridge tape. The tar **-r** command has to rewrite the last block on the tape, which the Exabyte hardware does not allow. Use the commands dump and dd to append. They work well, as they do not rewrite any blocks.

x extracts the named files 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 lists the names of the specified files each time they occur on the tape. If no file argument is given, all of the names on the tape are listed.

u adds the named files to the tape if either they are not already there or have been modified since last put on the tape.

c creates a new tape; writing begins on the beginning of the tape instead of after the last file. This command implies the **r** command letter.

o suppresses directory information. On output, tar normally places information specifying owner and modes of directories in the archive. Former versions of tar, when encountering this information, give error messages of the form

```
<name>/: cannot create
```

p restores files to their original modes, ignoring the present umask *(System Calls)*. Setuid and sticky information is also restored to the super-user.

MODIFIERS

0, ..., 9
selects an alternate drive on which the tape is mounted. The default is drive 0 at 6250 bpi, which is normally `/dev/rschs`.

v (verbose) types the name of each file preceded by the command letter. Normally tar does its work silently. With the t command letter, the verbose option gives more information about the tape entries than just their names.

w displays the action to be taken followed by file name, then wait for user confirmation. If a word beginning with y is given, the action is done. Any other input means do not do it.

f uses the next argument as the name of an archive. A hyphen specifies the standard output or the standard input, whichever is appropriate. Thus, tar can be used in a pipline to move hierarchies (see example below).

b uses the next argument as the blocking factor for tape records. The default is 20 512-byte blocks. This option should only be used with raw magnetic tape archives (see f above). The block size is determined automatically when reading tapes (command letters x and t). The recommended blocking factor for cartridge tapes is 126.

l complains if tar cannot resolve all of the links to the files dumped. If this is not specified, no error messages are displayed.

m disables restoration of modification times. The modification time is the time of extraction.

h follows symbolic links as if they were normal files or directories. Normally, tar does not follow symbolic links.

B forces input and output blocking to 20 blocks per record. This option was added so that tar can work across a communications channel where the blocking may not be maintained.

NOTES

- When extracting tapes created with the –r or –u options, directory modification times may not be set correctly.
- Previous restrictions dealing with tar's inability to properly handle blocked archives have been lifted.
- 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 file name length is 100 characters.
- There is no way to selectively follow symbolic links.

EXAMPLES

To archive files from /usr/include and from /etc, one might use:

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

Tar can be used to move hierarchies with a command such as:

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

DIAGNOSTICS

- Complaints about bad key characters and tape read/write errors.
- Complaints if enough memory is not available to hold the link tables.

FILES

/tmp/tar*

tbl

format tables for nroff or troff (unsupported)

The tbl command invokes a preprocessor for formatting tables for nroff or troff. The input files are processed and copied to the standard output.

---

tbl [*file...*]

ARGUMENTS

file specifies an input file. If no arguments are given, *tbl* reads the standard input, so it can be used as a filter.

NOTES

- When tbl is used with eqn or neqn the tbl command should be first, to minimize the volume of data passed through pipes.

SEE ALSO

nroff, troff, eqn
Document Preparation Guide: TBL – A Program to Format Tables

tc

phototypesetter simulator

The tc command interprets device codes for a Graphic Systems phototypesetter (cat). The standard output of tc is intended for a Tektronix 4015 (a 4014 terminal with ASCII and APL character sets).

---

tc [–t] [–s*n*] [–p*l*] [*file*]

ARGUMENTS

–t disables waiting between pages (for directing output into a file).

–s*n* skips the first *n* pages.

–p*l* sets the page length to *l*, which can include the scale factors:

p	points
i	inches
c	centimeters
P	picas (picas per page = lines * 47) (default)

'–*l w*' multiplies the default aspect ratio, 1.5, of a displayed page by *l*/*w*.

file specifies an input file (the default is the standard input).

DESCRIPTION

The sixteen typesetter sizes are mapped into the 4014's four sizes; the entire *troff* character set is drawn using the 4014's character generator, using overstruck combinations where necessary. Typical usage:

```
troff –t file | tc
```

At the end of each page tc waits for a newline (empty line) from the keyboard before continuing on to the next page. In this wait state, the command e suppresses the screen erase before the next page; s*n* causes the next *n* pages to be skipped; and !line sends line to the shell.

NOTES

- Font distinctions are lost.
- Tc's character set is limited to ASCII in just one size.

SEE ALSO

troff, plot

tcopy

make a copy of a magnetic tape, or print information about tape format

Tcopy may be used to copy a magnetic tape, or to print information about a tape's format. The tape may be in just about any format, as long as it has two tape marks at the end.

---

tcopy *source* [*destination*]

ARGUMENTS

source	name of source tape drive
destination	name of destination tape drive

DESCRIPTION

Tcopy copies a magnetic tape and prints information about its format. The only assumption made about the tape is that there are two tape marks at the end.

If you specify both a source tape drive and a destination tape drive, tcopy copies the source tape to the destination. The blocking on the destination tape is identical to that used on the source tape. Tcopy will also output (to standard output) information about the sizes of the tape files and records.

If you specify only a source tape, tcopy reads the tape and outputs information about the sizes of the tape files and records.

SEE ALSO

Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: mtio

tcpdump

dump traffic on a network

Tcpdump prints out the headers of packets on a network interface that match the boolean expression. Access is controlled by the permissions on /dev/bpf0, etc. By default, you must be root to invoke tcpdump.

---

tcpdump [-deflnNOpqStvx] [-c *count*] [-F *file*]

tcpdump [-i *interface*] [-r *file*] [-s *snaplen*]

tcpdump [-w *file*] *expression*

ARGUMENTS

-c Exit after receiving *count* packets.

-d Dump the compiled packet-matching code to standard output and stop.

-e Print the link-level header on each dump line.

-f Print 'foreign' internet addresses numerically rather than symbolically (useful if you are using the Network Information Service ypserv.

-F Use *file* as input for the filter expression. An additional expression given on the command line is ignored.

-i Listen on interface. If unspecified, tcpdump searches the system interface list for the lowest numbered, configured up interface (excluding loopback). Ties are broken by choosing the earliest match.

-l Make stdout line buffered. Useful if you want to see the data while capturing it. See the following examples:

```
% tcpdump -l | tee dat

% tcpdump -l  > dat & tail -f dat
```

-n Don't convert addresses (i.e., host addresses, port numbers, etc.) to names.

-N Don't print domain name qualification of host names. For example, if you give this flag then tcpdump will print "nic" instead of "nic.ddn.mil".

-O Do not run the packet-matching code optimizer. This is useful only if you suspect a bug in the optimizer.

-p Don't put the interface into promiscuous mode. (Promiscuous mode means that all packets will be received.) Note that the interface might be in promiscuous mode for some other reason; hence, -p cannot be used as an abbreviation for 'ether host {localhost} or broadcast'.

-q Quick (quiet?) output. Print less protocol information so output lines are shorter.

-r Read packets from *file* (which was created with the -w option). Standard input is used if file is "-".

-s Snarf *snaplen* bytes of data from each packet rather than the default of 68. 68 bytes is adequate for IP, ICMP,TCP and UDP but may truncate protocol information from name server and NFS packets (see below). Packets truncated because of a limited snapshot are indicated in the output with "[|]". Note that taking larger snapshots both increases the amount of time it takes to process packets and, effectively, decreases the amount of packet buffering. This may cause packets to be lost. You should limit *snaplen* to the smallest number that will capture the protocol information you're interested in.

-S Print absolute, rather than relative, TCP sequence numbers.

-t Don't print a timestamp on each dump line.

-v (Slightly more) verbose output. For example, the time to live and type of service information in an IP packet is printed.

-w Write the raw packets to *file* rather than parsing and printing them out. They can later be printed with the -r option. Standard output is used if *file* is "-".

-x Print each packet (minus its link level header) in hex. The smaller of the entire packet or *snaplen* bytes will be printed.

expression
selects which packets will be dumped. If no *expression* is given, all packets on the net will be dumped. Otherwise, only packets for which *expression* is 'true' will be dumped.

The *expression* consists of one or more *primitives*. Primitives usually consist of an *id* (name or number) preceded by one or more qualifiers. There are three different kinds of qualifier:

type qualifiers say what kind of thing the *id* name or number refers to. Possible types are *net* and *host;* e.g., 'host foo', 'net 128.3', 'port 20'. If there is no *type* qualifier, *host* is assumed.

dir qualifiers specify a particular tranfer direction to and/or from *id*. Possible directions are *src or dst*; e.g., 'src foo', 'dst net 128.3', 'src or dst port ftp-data'. If there is no *dir* qualifier, *src or dst* is assumed.

protoqualifiers restrict the match to a particular protocol. Possible protos are: *tcp* and *ether*; e.g., 'ether src foo', 'arp net 128.3', 'tcp port 21'. If there is no *proto* qualifier, all protocols consistent with the type are assumed. E.g., 'src foo' means '(ip or arp or rarp) src foo' (except the latter is not legal syntax), 'net bar' means '(ip or arp or rarp) net bar' and 'port 53' means '(tcp or udp) port 53'.

In addition to the above, there are some special 'primitive' keywords that don't follow the pattern: *greater* and arithmetic expressions. All of these are described below.

More complex filter expressions are built up by using the words *or* and *not* to combine primitives (for example,'host foo and not port ftp and not port ftp-data'). To save typing, identical qualifier lists can be omitted. For example,'tcp dst port ftp or ftp-data or domain' is exactly the same as 'tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain'.

Allowable primitives include:

dst host *host*
: True if the IP destination field of the packet is *host*, which may be either an address or a name.

src host *host*
: True if the IP source field of the packet is *host*.

host *host*
: True if either the IP source or destination of the packet is *host*. Any of the above host expressions can be prepended with the keywords, **ip, arp, or rarp as in:**

```
ip host host
```

which is equivalent to:

```
ether proto \ip and host host
```

if *host* is a name with multiple IP addresses, each address will be checked for a match.

ether dst *ehost*
: True if the ethernet destination address is *ehost*. *ehost* may be either a name from /etc/ethers or a number (see below for numeric format).

ether src *ehost*
: True if the ethernet source address is *ehost*.

ether host *ehost*
: True if either the ethernet source or destination address is *ehost*.

gateway *host*
: True if the packet used *host* as a gateway; i.e., the ethernet source or destination address was *host* but neither the IP source nor the IP destination was *host*. *Host* must be a name and must be found in both /etc/hosts and /etc/ethers. (An equivalent expression is

```
ether host ehost and not host host
```

which can be used with either names or numbers for *host / ehost.)*

dst net *net*
: True if the IP destination address of the packet has a network number of *net*, which may be either an address or a name.

src net *net*
: True if the IP source address of the packet has a network number of *net*.

net *net*
: True if either the IP source or destination address of the packet has a network number of *net*.

dst port *port*
: True if the packet is ip/tcp or ip/udp and has a destination port value of *port*. The *port* can be a number or a name used in /etc/services. If a name is used, both the port number and protocol are checked. If a number or ambiguous name is used, only the port number is checked (e.g., **dst port 513** will print both tcp/login traffic and udp/who traffic, and **port domain** will print both tcp/domain and udp/domain traffic).

src port *port*
: True if the packet has a source port value of *port*.

port *port*
: True if either the source or destination port of the packet is *port*. Any of the above port expressions can be prepended with the keywords, **tcp** or **udp**, as in the following example, which matches only tcp packets:

```
tcp src port port
```

less *length*
: True if the packet has a length less than or equal to *length*. This is equivalent to:

```
len <= length.
```

greater *length*
: True if the packet has a length greater than or equal to *length*. This is equivalent to:

```
len >= length.
```

ip proto *protocol*
: True if the packet is an ip packet of protocol type *protocol*. *Protocol* can be a number or one of the names *icmp, udp, nd,* or *tcp*. Note that the identifiers **tcp** and **udp** are also keywords and must be escaped via backslash (\), which is \\ in the C shell.

broadcast
: True if the packet is a broadcast packet.

ether proto *protocol*
: True if the packet is of ether type *protocol*. Protocol can be a number or a name like **ip, arp,** or **rarp**. Note these identifiers are also keywords and must be escaped via backslash (\).

ip, arp, rarp
Abbreviations for:

```
ether proto p
```

where *p* is one of the above protocols.

tcp, udp
Abbreviations for:

```
ip proto p
```

where *p* is one of the above protocols.

expr relop expr
True if the relation holds, where *relop* is one of >, <, >=, <=, =, !=, and *expr* is an arithmetic expression composed of integer constants (expressed in standard C syntax), the normal binary operators [+, −, *, /, &, |], a length operator, and special packet data accessors. To access data inside the packet, use the following syntax:

```
proto [ expr : size ]
```

Proto is one of *ether, ip, arp, rarp, tcp,* or *udp*, and indicates the protocol layer for the index operation. The byte offset, relative to the indicated protocol layer, is given by *expr*. *Size* is optional and indicates the number of bytes in the field of interest; it can be one, two, or four, and defaults to one. The length operator, indicated by the keyword **len**, gives the length of the packet.

For example, 'ether[0] & 1 != 0' catches all multicast traffic. The expression 'ip[0] & 0xf != 5' catches all IP packets with options. The expression 'ip[2:2] & 0x1fff = 0' catches only unfragmented datagrams and frag zero of fragmented datagrams. This check is implicitly applied to the tcp and udp index opertations. For instance, tcp[0] always means the first byte of the TCP *header*, and never means the first byte of an intervening fragment.

Primitives may be combined using:

A parenthesized group of primitives and operators (parentheses are special to the Shell and must be escaped).

Negation ('!' or 'not').

Concatenation ('and').

Alternation ('or').

Negation has highest precedence. Alternation and concatenation have equal precedence and associate left to right. Note that explicit and tokens, not juxtaposition, are now required for concatenation.

If an identifier is given without a keyword, the most recent keyword is assumed. For example,

```
not host vs and ace
```

is short for

```
not host vs and host ace
```

which should not be confused with

```
not ( host vs or ace )
```

Expression arguments can be passed to tcpdump as either a single argument or as multiple arguments, whichever is more convenient. Generally, if the expression contains Shell metacharacters, it is easier to pass it as a single, quoted argument. Multiple arguments are concatenated with spaces before being parsed.

EXAMPLES

To print all packets arriving at or departing from *sundown:*

```
tcpdump host sundown
```

To print traffic between *helios* and either *hot* or *ace:*

```
tcpdump host helios and \( hot or ace \)
```

To print all IP packets between *ace* and any host except *helios:*

```
tcpdump ip host ace and not helios
```

To print all traffic between local hosts and hosts at Berkeley:

```
tcpdump net ucb-ether
```

To print all ftp traffic through internet gateway *snup:* (Note that the expression is quoted to prevent the shell from (mis-)interpreting the parentheses):

```
tcpdump 'gateway snup and (port ftp or ftp-data)'
```

To print traffic neither sourced from nor destined for local hosts (if you gateway to one other net, this stuff should never make it onto your local net).

```
tcpdump ip and not net localnet
```

To print the start and end packets (the SYN and FIN packets) of each TCP conversation that involves a non-local host.

```
tcpdump 'tcp[13] & 3 != 0 and not src and dst net localnet'
```

To print IP packets longer than 576 bytes sent through gateway *snup:*

```
tcpdump 'gateway snup and ip[2:2] > 576'
```

To print IP broadcast or multicast packets that were not sent via ethernet broadcast or multicast:

```
tcpdump 'ether[0] & 1 = 0 and ip[16] >= 224'
```

Link Level Headers

If the -e option is given, the link level header is printed out. On ethernets, the source and destination addresses, protocol, and packet length are printed.

> NOTE: The following description assumes familiarity with the SLIP compression algorithm described in RFC-1144.

On SLIP links, a direction indicator ("I" for inbound, "O" for outbound), packet type, and compression information are printed out. The packet type is printed first. The three types are *ip*, *utcp*, and *ctcp*. No further link information is printed for ip packets. For TCP packets, the connection identifier is printed following the type. If the packet is compressed, its encoded header is printed out. The special cases are printed out as *S+n and *SA+n, where *n* is the amount by which the sequence number (or sequence number and ack) has changed. If it is not a special case, zero or more changes are printed. A change is indicated by U(urgent pointer), W (window), A (ack), S (sequence number), and I (packet ID), followed by a delta (+n or -n), or a new value (=n). Finally, the amount of data in the packet and compressed header length are printed.

For example, the following line shows an outbound compressed TCP packet, with an implicit connection identifier; the ack has changed by 6, the sequence number by 49, and the packet ID by 6; there are 3 bytes of data and 6 bytes of compressed header:

```
O ctcp * A+6 S+49 I+6 3 (6)
```

ARP/RARP Packets

Arp/rarp output shows the type of request and its arguments. The format is intended to be self explanatory. Here is a short sample taken from the start of an rlogin from host rtsg to host csam:

```
arp who-has csam tell rtsg
arp reply csam is-at CSAM
```

The first line says that rtsg sent an arp packet asking for the ethernet address of internet host csam. Csam replies with its ethernet address (in this example, ethernet addresses are in caps and internet addresses in lower case).

This would look less redundant if we had done tcpdump -n:

```
arp who-has 128.3.254.6 tell 128.3.254.68
arp reply 128.3.254.6 is-at 02:07:01:00:01:c4
```

If we had done tcpdump -e, the fact that the first packet is broadcast and the second is point-to-point would be visible:

```
RTSG Broadcast 080664: arp who-has csam tell rtsg
CSAM RTSG 0806 64: arp reply csam is-at CSAM
```

For the first packet this says the ethernet source address is RTSG, the destination is the broadcast address, the type field contained hex 0806 (type ETHER_ARP) and the total length was 64 bytes.

TCP Packets

> **NOTE:** The following description assumes familiarity with the TCP protocol described in RFC–793. If you are not familiar with the protocol, neither this description nor tcpdump will be of much use to you.

The general format of a tcp protocol line is:

```
src > dst: flags data-seqno ack window urgent options
```

Src and *dst* are the source and destination IP addresses and ports. *Flags* are some combination of S (SYN), F (FIN), P (PUSH) or R (RST) or a single '.' (no flags). *Data-seqno* describes the portion of sequence space covered by the data in this packet (see example below). *Ack* is the sequence number of the next data expected from the other direction on this connection. *Window* is the number of bytes of receive buffer space available from the other direction on this connection. *Urg* indicates there is 'urgent' data in the packet. Options are *tcp* options enclosed in angle brackets (e.g., <mss 1024>).

Src, dst and *flags* are always present. The other fields depend on the contents of the packet's tcp protocol header and are output only if appropriate.

Here is the opening portion of an rlogin from host rtsg to host csam.

```
rtsg.1023 > csam.login: S 768512:768512(0) win 4096 <mss 1024>
csam.login > rtsg.1023: S 947648:947648(0) ack 768513 win 4096 <mss 1024>
rtsg.1023 > csam.login: . ack 1 win 4096
rtsg.1023 > csam.login: P 1:2(1) ack 1 win 4096
csam.login > rtsg.1023: . ack 2 win 4096
rtsg.1023 > csam.login: P 2:21(19) ack 1 win 4096
csam.login > rtsg.1023: P 1:2(1) ack 21 win 4077
csam.login > rtsg.1023: P 2:3(1) ack 21 win 4077 urg 1
csam.login > rtsg.1023: P 3:4(1) ack 21 win 4077 urg 1
```

The first line says that tcp port 1023 on rtsg sent a packet to port login on csam. The S indicates that the SYN flag was set. The packet sequence number was 768512 and it contained no data. (The notation is'first:last(nbytes)' which means 'sequence numbers first up to but not including last which is nbytes bytes of user data'.)There was no piggy-backed ack, the available receive window was 4096 bytes and there was a max-segment-size option requesting an mss of 1024 bytes.

Csam replies with a similar packet except it includes a piggy-backed ack for rtsg's SYN. Rtsg then acks csam's SYN. The '.' means no flags were set. The packet contained no data so there is no data sequence number. Note that the ack sequence number is a small integer (1). The first time tcpdump sees a tcp 'conversation',it prints the sequence number from the packet. On subsequent packets of the conversation, the difference between the current packet's sequence number and this initial sequence number is printed. This means that sequence numbers after the first can be interpreted as relative byte positions in the conversation's data stream (with the first data byte each direction being '1'). '–S' will override this feature, causing the original sequence numbers to be output.

On the 6th line, rtsg sends csam 19 bytes of data (bytes 2 through 20 in the rtsg -> csam side of the conversation). The PUSH flag is set in the packet. On the 7th line, csam says it's received data sent by rtsg up to but not including byte 21. Most of this data is apparently sitting in the socket buffer since csam's receive window has gotten 19 bytes smaller. Csam also sends one byte of data to rtsg in this packet. On the 8th and 9th lines, csam sends two bytes of urgent, pushed data to rtsg.

UDP Packets

UDP format is illustrated by this rwho packet:

```
actinide.who > broadcast.who: udp 84
```

This says that port who on host actinide sent a udp datagram to port who on host broadcast, the Internet broadcast address. The packet contained 84 bytes of user data.

Some UDP services are recognized (from the source or destination port number) and the higher level protocol information printed. In particular, Domain Name service requests (RFC-1034/1035) and Sun RPC calls (RFC-1050) to NFS.

UDP Name Server Requests

> **NOTE:** The following description assumes familiarity with the Domain Service protocol described in RFC-1035.

Name server requests are formatted as

```
src > dst: id op? flags qtype qclass name (len)
h2opolo.1538 > helios.domain: 3+ A? ucbvax.berkeley.edu. (37)
```

Host h2opolo asked the domain server on helios for an address record (qtype=A) associated with the name ucbvax.berkeley.edu. The query id was '3'. The '+' indicates the recursion desired flag was set. The query length was 37 bytes, not including the UDP and IP protocol headers. The query operation was the normal one, Query, so the *op* field was omitted. If the *op* had been anything else, it would have been printed between the '3' and the '+'. Similarly, the *qclass* was the normal one, C_IN, and omitted. Any other qclass would have been printed immediately after the 'A'.

A few anomalies are checked and may result in extra fields enclosed in square brackets: If a query contains an answer, name server or authority section, or arcount are printed as '[na]', '[nn]' or '[nau]' where *n* is the appropriate count. If any of the response bits are set (AA, RA or rcode) or any of the 'must be zero' bits are set in bytes two and three, '[b2&3=x]' is printed, where *x* is the hex value of header bytes two and three.

UDP Name Server Responses

Name server responses are formatted as

```
src > dst: id op rcode flags a/n/au type class data (len)
helios.domain > h2opolo.1538: 3 3/3/7 A 128.32.137.3 (273)
helios.domain > h2opolo.1537: 2 NXDomain* 0/1/0 (97)
```

In the first example, helios responds to query id 3 from h2opolo with 3 answer records, 3 name server records and 7 authority records. The first answer record is type A (ad-

dress) and its data is internet address 128.32.137.3. The total size of the response was 273 bytes, excluding UDP and IP headers. The op (Query) and response code (NoError) were omitted, as was the class (C_IN) of the A record.

In the second example, helios responds to query 2 with a response code of non-existent domain (NXDomain) with no answers, one name server and no authority records. The '*' indicates that the authoritative answer bit was set. Since there were no answers, no type, class or data were printed.

Other flag characters that might appear are '-' (recursion available, RA, not set) and '|' (truncated message, TC, set). If the 'question' section doesn't contain exactly one entry, '[nq]' is printed.

Note that name server requests and responses tend to be large and the default *snaplen* of 96 bytes may not capture enough of the packet to print. Use the -s flag to increase the snaplen if you need to seriously investigate name server traffic. '-s 128' is a good number.

NFS Requests

Sun NFS (Network File System) requests and replies are printed as:

```
src.xid > dst.nfs: len op args
src.nfs > dst.xid: reply stat len
vs.e2766 > helios.nfs: 136 readdir fh 6.5197 8192 bytes @ 0
helios.nfs > vs.e2766: reply ok 384
vs.e2767 > helios.nfs: 136 lookup fh 6.5197 `RCS'
```

In the first line, host vs sends a transaction with id e2766 to helios (note that the number following the src host is a transaction id, not the source port). The request was 136 bytes, excluding the UDP and IP headers. The operation was a readdir (read directory) on file handle (fh) 6.5197. 8192 bytes are read, starting at offset 0. Helios replies 'ok' with 384 bytes of data. (The design of Sun's RPC protocol makes it difficult to interpret replies.)

In the third line, vs asks helios to lookup the name 'RCS' in directory file 6.5197. Note that the data printed depends on the operation type. The format is intended to be self explanatory if read in conjunction with an NFS protocol spec. (See the *Concentrix Network File System Manual,* chapter 6.)

Note that NFS requests are very large and the above won't be printed unless *snaplen* is increased. You can use '-s 192' to watch NFS traffic.

KIP Appletalk (DDP in UDP)

Appletalk DDP packets encapsulated in UDP datagrams are de-encapsulated and dumped as DDP packets (i.e., all the UDP header information is discarded). The file /etc/atalk.names is used to translate appletalk net and node numbers to names. Lines in this file have the form

```
number          name
1.254           ether
16.1            icsd-net
1.254.110       ace
```

The first two lines give the names of appletalk networks. The third line gives the name of a particular host (a host is distinguished from a net by the 3rd octet in the number – a net number must have two octets and a host number must have three octets.) The number and name should be separated by whitespace (blanks or tabs). The /etc/atalk.names file may contain blank lines or comment lines (lines starting with a '#').

Appletalk addresses are printed in the form

```
net.host.port

144.1.209.2 > icsd-net.112.220
office.2 > icsd-net.112.220
jssmag.149.235 > icsd-net.2
```

(If the /etc/atalk.names doesn't exist or doesn't contain an entry for some appletalk host/net number, addresses are printed in numeric form.) In the first example, NBP(DDP port 2) on net 144.1 node 209 is sending to whatever is listening on port 220 of net icsd node 112. The second line is the same except the full name of the source node is known ('office'). The third line is a send from port 235 on net jssmag node149 to broadcast on the icsd-net NBP port (note that the broadcast address (255) is indicated by a net name with no host number – for this reason it's a good idea to keep node names and net names distinct in /etc/atalk.names).

NBP (name binding protocol) and ATP (Appletalk transaction protocol) packets have their contents interpreted. Other protocols just dump the protocol name (or number if no name is registered for the protocol) and packet size.

NBP packets are formatted like the following examples:

```
icsd-net.112.220 > jssmag.2: nbp-lkup 190: "=:LaserWriter@*"
jssmag.209.2 > icsd-net.112.220: nbp-reply 190: "RM1140:LaserWriter@*" 250
techpit.2 > icsd-net.112.220: nbp-reply 190: "techpit:LaserWriter@*" 186
```

The first line is a name lookup request for laserwriters sent by net icsd host 112 and broadcast on net jssmag. The nbp id for the lookup is 190. The second line shows a reply for this request (note that it has the same id) from host jssmag.209 saying that it has a laserwriter resource named "RM1140" registered on port 250. The third line is another reply to the same request saying host techpit has laserwriter "techpit" registered on port 186.

ATP packet formatting is demonstrated by the following example:

```
jssmag.209.165 > helios.132: atp-req  12266<0-7> 0xae030001
helios.132 > jssmag.209.165: atp-resp 12266:0 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:1 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:2 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:3 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:4 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:5 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:6 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp*12266:7 (512) 0xae040000
jssmag.209.165 > helios.132: atp-req  12266<3,5> 0xae030001
helios.132 > jssmag.209.165: atp-resp 12266:3 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:5 (512) 0xae040000
jssmag.209.165 > helios.132: atp-rel  12266<0-7> 0xae030001
jssmag.209.133 > helios.132: atp-req* 12267<0-7> 0xae030002
```

Jssmag.209 initiates transaction id 12266 with host helios by requesting up to 8 packets (the '<0-7>'). The hex number at the end of the line is the value of the 'userdata' field in the request.

Helios responds with 8 512-byte packets. The ':digit' following the transaction id gives the packet sequence number in the transaction and the number in parens is the amount of data in the packet, excluding the atp header. The '*' on packet 7 indicates that the EOM bit was set.

Jssmag.209 then requests that packets 3 & 5 be retransmitted. Helios resends them, then jssmag.209 releases the transaction. Finally, jssmag.209 initiates the next request. The '*' on the request indicates that XO ('exactly once') was not set.

IP Fragmentation

Fragmented Internet datagrams are printed as

```
(frag id:size@offset+)
(frag id:size@offset)
```

The first form indicates there are more fragments. The second indicates this is the last fragment.

Id is the fragment id (in hex). *Size* is the fragment size (in bytes) excluding the IP header. *Offset* is this fragment's offset (in bytes) in the original datagram.

The fragment information is output for each fragment. The first fragment contains the higher level protocol header and the frag information is printed after the protocol information. Fragments after the first contain no higher level protocol header and the frag information is printed after the source and destination addresses. For example, here is part of an ftp from arizona.edu to lbl-rtsg.arpa over a CSNET connection that doesn't appear to handle 576 byte datagrams:

```
arizona.ftp-data > rtsg.1170: . 1024:1332(308) ack 1 win 4096 (frag 595a:328@0+)
arizona > rtsg: (frag 595a:204@328)
rtsg.1170 > arizona.ftp-data: . ack 1536 win 2560
```

There are a couple of things to note here: First,addresses in the 2nd line don't include port numbers. This is because the TCP protocol information is all in the first fragment and we have no idea what the port or sequence numbers are when we print the later fragments. Second, the tcp sequence information in the first line is printed as if there were 308 bytes of user data when, in fact, there are 512 bytes (308 in the first frag and 204 in the second). If you are looking for holes in the sequence space or trying to match up acks with packets, this can fool you. A packet with the IP don't fragment flag is marked with a trailing (DF).

Timestamps

By default, all output lines are preceded by a timestamp. The timestamp is the current clock time in the form

```
hh:mm:ss.frac
```

and is accurate to 10 msec (same as the kernel's clock). The timestamp reflects the time the kernel first saw the packet. No attempt is made to account for the time lag between when the ethernet interface removed the packet from the wire and when the kernel serviced the 'new packet' interrupt .

NOTES

- Some attempt should be made to reassemble IP fragments or, at least to compute the right length for the higher level protocol.
- Name server inverse queries are not dumped correctly: The (empty) question section is printed rather than real query in the answer section. Some believe that inverse queries are themselves a bug and prefer to fix the program generating them rather than tcpdump.
- Apple Ethertalk DDP packets could be dumped as easily as KIP DDP packets but aren't.
- A packet trace that crosses a daylight savings time change will give skewed time stamps (the time change is ignored).

SEE ALSO

Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: bpf

tee

copy to standard output and to file

The **tee** command copies the standard input to the standard output and the specified files. By default, **tee** overwrites the output files and responds to interrupts.

---

tee [-i] [-a] [*file*...]

ARGUMENTS

-i ignores interrupts.

-a appends to the specified files rather than overwriting them.

file specifies an output file.

telnet

user interface to the TELNET protocol

The telnet command is used to communicate with another host using the TELNET protocol.

---

telnet [*host* [*port*]]

ARGUMENTS

If telnet is invoked with arguments, it performs an open command (see below) with those arguments.

If telnet is invoked without arguments, it enters command mode, indicated by its prompt `telnet>`. In this mode, it accepts and executes the commands listed below.

DESCRIPTION

Once a connection has been opened, telnet enters an input mode. The input mode entered will be either "character at a time" or "line by line" depending on what the remote system supports.

In "character at a time" mode, most text typed is immediately sent to the remote host for processing.

In "line by line" mode, all text is echoed locally, and (normally) only completed lines are sent to the remote host. The "local echo character" (initially "CTRL E") may be used to turn off and on the local echo (this would mostly be used to enter passwords without the password being echoed).

In either mode, if the *localchars* toggle is TRUE (the default in line mode; see below), the user's quit, intr, and flush characters are trapped locally, and sent as TELNET protocol sequences to the remote side. There are options (see toggle autoflush and toggle autosynch below) which cause this action to flush subsequent output to the terminal (until the remote host acknowledges the TELNET sequence) and flush previous terminal input (in the case of quit and intr).

While connected to a remote host, telnet command mode may be entered by typing the telnet "escape character" (initially "CTRL]"). When in command mode, the normal terminal editing conventions are available.

If the remote server supports the 'Negotiate About Window Size' option, telnet will respond to this protocol and send update messages to the server when it is notified of window size changes by the SIGWINCH signal.

COMMANDS

The following commands are available. Only enough of each command to uniquely identify it need be typed. This is also true for arguments to the mode, set, toggle, and display commands.

open *host* [*port*]

Open a connection to the named *host*. If no port number is specified, *telnet* will attempt to contact a TELNET server at the default port. The *host* specification may be either a host name (see hosts(5)) or an Internet address specified in the "dot notation" (see inet(3)).

close Close a TELNET session and return to command mode.

quit Close any open TELNET session and exit telnet. An end of file (in command mode) will also close a session and exit.

z Suspend telnet. This command only works when the user is using the csh.

mode *type*

Type is either line (for "line by line" mode) or character (for "character at a time" mode). The remote host is asked for permission to go into the requested mode. If the remote host is capable of entering that mode, the requested mode will be entered.

status Show the current status of telnet. This includes the peer one is connected to, as well as the current mode.

display [*argument...*]

Displays all, or some, of the set and toggle values (see below).

? [*command*]

Get help. With no arguments, telnet prints a help summary. If a command is specified, telnet will print the help information for just that command.

send *arguments*

Sends one or more special character sequences to the remote host. The following are the arguments which may be specified (more than one argument may be specified at a time):

escape Sends the current telnet escape character (initially "CTRL]").

synch Sends the TELNET SYNCH sequence. This sequence causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data (and may not work if the remote system is a 4.2 BSD system –if it doesn't work, a lower case "r" may be echoed on the terminal).

brk Sends the TELNET BRK (Break) sequence, which may have significance to the remote system.

ip Sends the TELNET IP (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.

ao Sends the TELNET AO (Abort Output) sequence, which should cause the remote system to flush all output from the remote system to the user's terminal.

ayt Sends the TELNET AYT (Are You There) sequence, to which the remote system may or may not choose to respond.

ec Sends the TELNET EC (Erase Character) sequence, which should cause the remote system to erase the last character entered.

el Sends the TELNET EL (Erase Line) sequence, which should cause the remote system to erase the line currently being entered.

ga Sends the TELNET GA (Go Ahead) sequence, which likely has no significance to the remote system.

nop Sends the TELNET NOP (No OPeration) sequence.

? Prints out help information for the send command.

set *argument value*

Set any one of a number of telnet variables to a specific value. The special value "off" turns off the function associated with the variable. The values of variables may be interrogated with the display command. The variables which may be specified are:

echo This is the value (initially "CTRL E") which, when in "line by line" mode, toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for entering, say, a password).

escape This is the telnet escape character (initially "CTRL [") which causes entry into telnet command mode (when connected to a remote system).

interrupt If telnet is in localchars mode (see toggle *localchars* below) and the interrupt character is typed, a TELNET IP sequence (see send ip above) is sent to the remote host. The initial value for the interrupt character is taken to be the terminal's intr character.

quit If telnet is in localchars mode (see toggle *localchars* below) and the quit character is typed, a TELNET BRK sequence (see send brk above) is sent to the remote host. The initial value for the quit character is taken to be the terminal's quit character.

flushoutput If telnet is in localchars mode (see toggle *localchars* below) and the flushoutput character is typed, a TELNET AO sequence (see send ao above) is sent to the remote host. The initial value for the flush character is taken to be the terminal's flush character.

erase If telnet is in localchars mode (see toggle *localchars* below), and if *telnet* is operating in "character at a time" mode, then when this character is typed, a TELNET EC sequence (see *send ec* above) is sent to the remote system. The initial value for the erase character is taken to be the terminal's erase character.

kill If telnet is in localchars mode (see toggle *localchars* below), and if telnet is operating in "character at a time" mode, then when this charac-

ter is typed, a TELNET EL sequence (see **send el** above) is sent to the remote system. The initial value for the kill character is taken to be the terminal's kill character.

eof If **telnet** is operating in "line by line" mode, entering this character as the first character on a line will cause this character to be sent to the remote system. The initial value of the eof character is taken to be the terminal's eof character.

toggle *arguments...*

Toggle (between TRUE and FALSE) various flags that control how *telnet* responds to events. More than one argument may be specified. The state of these flags may be interrogated with the display command. Valid arguments are:

localchars If this is TRUE, then the flush, interrupt, quit, erase, and kill characters (see **set** above) are recognized locally, and transformed into (hopefully) appropriate TELNET control sequences (respectively ao, ip, brk, ec, and el; see *send* above). The initial value for this toggle is TRUE in "line by line" mode, and FALSE in "character at a time" mode.

autoflush If autoflush and localchars are both TRUE, then when the ao, intr, or quit characters are recognized (and transformed into TELNET sequences; see set above for details), *telnet* refuses to display any data on the user's terminal until the remote system acknowledges (via a TELNET Timing Mark option) that it has processed those TELNET sequences. The initial value for this toggle is TRUE if the terminal user had not done an "stty noflsh", otherwise FALSE (see *stty*).

autosynch If autosynch and localchars are both TRUE, then when either the intr or quit characters is typed (see *set* above for descriptions of the intr and quit characters), the resulting TELNET sequence sent is followed by the TELNET SYNCH sequence. This procedure should cause the remote system to begin throwing away all previously typed input until both of the TELNET sequences have been read and acted upon. The initial value of this toggle is FALSE.

crmod Toggle carriage return mode. When this mode is enabled, most carriage return characters received from the remote host will be mapped into a carriage return followed by a line feed. This mode does not affect those characters typed by the user, only those received from the remote host. This mode is not very useful unless the remote host only sends carriage return, but never line feed. The initial value for this toggle is FALSE.

debug Toggles socket level debugging (useful only to the superuser). The initial value for this toggle is FALSE.

options Toggles the display of some internal *telnet* protocol processing (having to do with TELNET options). The initial value for this toggle is FALSE.

netdata Toggles the display of all network data (in hexadecimal format). The initial value for this toggle is FALSE.

? Displays the legal toggle commands.

NOTES

- There is no adequate way for dealing with flow control.
- On some remote systems, echo has to be turned off manually when in "line by line" mode.
- There is enough settable state to justify a .telnetrc file.
- No capability for a .telnetrc file is provided.
- In "line by line" mode, the terminal's eof character is only recognized (and sent to the remote system) when it is the first character on a line.

test

evaluate expression and return status

The **test** command evaluates an expression and returns an exit status of zero if its value is true, false otherwise.

---

test *expr*

ARGUMENTS

expr is an expression as described below. If not specified, **test** returns a non-zero exit.

DESCRIPTION

The following primitives are used to construct *expr*.

-r *file*	true if the file exists and is readable.
-w *file*	true if the file exists and is writable.
-f *file*	true if the file exists and is not a directory.
-d *file*	true if the file exists exists and is a directory.
-h*file*	true if the file exists and is a symbolic link.
-s *file*	true if the file exists and has a size greater than zero.
-t [*fildes*]	true if the open file whose file descriptor number is *fildes* (1 by default) is associated with a terminal device.
-z *s1*	true if the length of string *s1* is zero.
-n *s1*	true if the length of the string *s1* is nonzero.
s1 **=** *s2*	true if the strings *s1* and *s2* are equal.
s1 **!=** *s2*	true if the strings *s1* and *s2* are not equal.
s1	true if *s1* is not the null string.
n1 **-eq** *n2*	true if the integers *n1* and *n2* are algebraically equal. Any of the comparisons -ne, -gt, -ge, -lt, or -le can be used in place of -eq.

The primitives can be combined with the following operators:

!	unary negation operator

-a	binary *and* operator
-o	binary *or* operator
(*expr*)	parentheses for grouping.

The binary *and* operator (-a) has higher precedence than the binary *or* operator (-o). All the operators and operands are separate arguments to **test**. Parentheses are meaningful to the shell and must be escaped.

SEE ALSO

sh, find

tftp

trivial file transfer program

Tftp is the user interface to the Internet TFTP (Trivial File Transfer Protocol), which allows users to transfer files to and from a remote machine.

---

tftp [*host*]

ARGUMENTS

host Name of the remote machine with which to transfer files. If the name is not given on the command line, it may be given after tftp is running (see below).

DESCRIPTION

Tftp is the user interface to the Internet TFTP (Trivial File Transfer Protocol), which allows users to transfer files to and from a remote machine. The remote host may be specified on the command line or after *tftp* is running.

Once tftp is running, it issues the prompt "tftp>" and recognizes the following commands:

connect *host-name* [*port*]

Set the host (and optionally port) for transfers. Note that the TFTP protocol, unlike the FTP protocol, does not maintain connections between transfers; thus, the connect command does not actually create a connection, but merely remembers what host is to be used for transfers. You do not have to use the *connect* command; the remote host can be specified as part of the get or put commands.

mode *transfer-mode*

Set the mode for transfers; *transfer-mode* may be one of "ascii" or "binary". The default is "ascii".

put *file*
put *localfile remotefile*
put *file1 file2 ... fileN remote-directory*

Put a file or set of files to the specified remote file or directory. The destination can be in one of two forms: 1) a filename on the remote host, if the host has already been specified; 2) a string of the form *host:filename* to specify both a host and filename at the same time. If the latter form is used, the host name specified becomes the default for future transfers.

If the remote-directory form is used, the remote host is assumed to be a UNIX machine.

get *filename*
get *remotename localname*
get *file1 file2 ... fileN*

Get a file or set of files from the specified sources. Source can be in one of two forms: 1) a filename on the remote host, if the host has already been specified; 2) a string of the form *host:filename* to specify both a host and filename at the same time. If the latter form is used, the last host name specified becomes the default for future transfers.

quit

Exit tftp. An end of file also exits.

verbose

Toggle verbose mode.

trace

Toggle packet tracing.

status

Show current status.

rexmt *retransmission-timeout*

Set the per-packet retransmission timeout, in seconds.

timeout *total-transmission-timeout*

Set the total transmission timeout, in seconds.

ascii

Shorthand for "mode ascii"

binary

Shorthand for "mode binary"

? [*command-name ...*]

Print help information.

NOTES

- The remote site will probably have some sort of file access restrictions in place. These file restrictions may affect your ability to copy certain files from the remote system, or to write files to the remote system.

time

time a command (C shell)

The time command executes the given command and, when completed, displays the time elapsed during the command. If necessary, an extra shell is created to print the time statistic when the command completes.

---

time [*command*]

ARGUMENTS

command is any Concentrix command. If omitted, a summary of time used by this shell and its children is printed.

DESCRIPTION

The times are displayed in the following format:

*x.xu y.ys hh:mm:ss tt% text+data*k *inblock+outblock*io *iii*pf+*jw*

x.x time in seconds spent executing for user.

y.y time in seconds spent in system executing on behalf of user.

hh:mm:ss wall clock time from beginning to end of process.

tt ((*x.x* + *y.y*) / ((*hh* * 60) + *mm*) * 60 + *ss*) * 100.

text average amount of unshared memory in kilobytes used for text region (see note below).

data average amount of unshared memory in kilobytes used for data region (see note below).

inblock number of 4K-byte blocks input by the file system on a BDS4.2 system; number of bytes on a track input by the file system on a track-allocated file system.

outblock number of 4K-byte blocks output by the file system on a BDS4.2 system; number of bytes on a track output by the file system on a track-allocated file system.

iii number of page faults.

j number of times process was swapped from main memory.

NOTES

- To time a command issued via the *execute* utility, use the following syntax:

```
% time execute [option...] command
```

- Time command output cannot be redirected in the normal way. For example:

```
% time foo > foo.time
```

places the standard output of `foo` in `foo.time`. To capture the standard output of *time*, create a shell script named, for example, `xtime` containing:

```
#
time $*
```

Execute the script, redirecting the output.

```
% xtime foo > foo.time
```

- Due to limitations in the internal representations, the *text* and *data* fields are unreliable for large jobs that run for a long time.

SEE ALSO

execute, time (/bin/time)

time

time a command

The time command executes the given command and, when completed, displays the time elapsed during the command, the time spent in the system, and the time spent in execution of the command. Times are reported in seconds.

/bin/time *command*

ARGUMENTS

command is any Concentrix command.

DESCRIPTION

The times are displayed on the diagnostic output stream.

NOTES

- Elapsed time is accurate to the second, while the CPU times are measured to the 100th second. Thus the sum of the CPU times can be up to a second larger than the elapsed time.
- To time a command issued via the execute utility, use the following syntax:

```
% /bin/time execute [option...] command
```

SEE ALSO

execute, time (C shell built-in)

times

print accumulated user and system times (Bourne Shell)

The times command prints the accumulated user and system times for processes run from the shell.

---

times

SEE ALSO

execute, /bin/time

tip

connect to a remote system

The tip command establishes a full-duplex connection to another machine, giving the appearance of being logged in directly on the remote cpu. You must have an account on the machine (or equivalent) to which you wish to connect.

---

tip [–v] [*–speed*] [*system-name*]

tip [–v] [*–speed*] *phone-number*

ARGUMENTS

–v causes tip to display set commands (without the `~s` prefix in a file .tiprc in your home directory) as they are made.

–speed specifies the desired baud rate, for example: `tip –300 mds`.

system-name specifies the name of a remote system. Tip uses the file /etc/remote to find how to reach a particular system and to find out how it should operate while talking to the system; refer to remote *(File Formats)* for a full description.

phone-number specifies the phone number of a remote system. When tip must dial a phone number to connect to a system it displays various messages indicating its actions. Tip supports the DEC DN-11 and Racal-Vadic 831 auto-call-units; the DEC DF02 and DF03, Ventel 212+, Racal-Vadic 3451, and Bizcomp 1031 and 1032 integral call unit/modems.

DESCRIPTION

Typed characters are normally transmitted directly to the remote machine (which does the echoing as well). A tilde (~) appearing as the first character of a line is an escape signal; the following are recognized:

~c [*name*] changes directory to the specifies name (the default is your home directory).

~CTRL**D** (also ~.) drops the connection and exits (you can still be logged in on the remote machine).

~p *from* [*to*] sends a file to a remote UNIX host. The put command causes the remote system to run the command string `cat > 'to'`, while tip sends it the `from` file. If the `to` file isn't specified the `from` file name is used. This command is actually a UNIX specific version of the `~>` command.

~s sets a variable (see the discussion below).

-t *from* [*to*] takes a file from a remote UNIX host. As in the put command, the to file defaults to the from file name if it isn't specified. The remote host executes the command string `cat 'from';echo ^A` to send the file to *tip*.

-CTRL Y stops only the local side of tip. The remote side of tip (that displays output from the remote host) is left running. This function is only available with job control.

-CTRL Z stops tip (only available with job control).

-! escapes to a shell (exiting the shell returns you to tip).

-> copies a file from local to remote. Tip prompts for the name of a local file to transmit.

-< copies a file from remote to local. Tip prompts first for the name of the file to be sent, then for a command to be executed on the remote machine.

-| pipes the output from a remote command to a local process. The command string sent to the local system is processed by the shell.

-# sends a BREAK to the remote system. For systems that do not support the necessary ioctl call the break is simulated by a sequence of line speed changes and DEL characters.

-$ pipes the output from a local process to the remote host. The command string sent to the local system is processed by the shell.

-? gets a summary of the tilde escapes

When tip establishes a connection it sends out a connection message to the remote system; the default value, if any, is defined in /etc/remote.

When tip prompts for an argument (e.g. during setup of a file transfer) the line typed can be edited with the standard erase and kill characters. A null line in response to a prompt, or an interrupt, aborts the dialogue and returns you to the remote machine.

Tip guards against multiple users connecting to a remote system by opening modems and terminal lines with exclusive access, and by honoring the locking protocol used by *uucp*.

During file transfers tip provides a running count of the number of lines transferred. When using the -> and -< commands, the *eofread* and *eofwrite* variables are used to recognize end-of-file when reading, and specify end-of-file when writing (see below). File transfers normally depend on tandem mode for flow control. If the remote system does not support tandem mode, echocheck can be set to indicate tip should synchronize with the remote system on the echo of each transmitted character.

VARIABLES

Tip maintains a set of *variables* that control its operation. Some of these variable are read-only to normal users (root is allowed to change anything of interest). Variables can

be displayed and set through the s escape. The syntax for variables is patterned after vi and Mail. Supplying `all` as an argument to the set command displays all variables readable by the user. Alternatively, the user can request display of a particular variable by attaching a question mark to the end. For example `escape?` displays the current escape character.

Variables are numeric, string, character, or boolean values. Boolean variables are set merely by specifying their name; they can be reset by prepending an exclamation point to the name. Other variable types are set by concatenating an equal sign and the value. The entire assignment must not have any blanks in it. A single set command can be used to interrogate as well as set a number of variables. Variables can be initialized at run time by placing set commands (without the `~s` prefix in a file .tiprc in one's home directory). The `-v` option causes tip to display the sets as they are made. Certain common variables have abbreviations. The following is a list of common variables, their abbreviations, and their default values.

baudrate (num) The baud rate at which the connection was established; abbreviated *ba*.

beautify (bool) Discard unprintable characters when a session is being scripted; abbreviated *be*.

dialtimeout (num) When dialing a phone number, the time (in seconds) to wait for a connection to be established; abbreviated *dial*.

echocheck (bool) Synchronize with the remote host during file transfer by waiting for the echo of the last character transmitted; default is *off*.

eofread (str) The set of characters that signify an end-of-transmission during a `~<` file transfer command; abbreviated *eofr*.

eofwrite (str) The string sent to indicate end-of-transmission during a `->` file transfer command; abbreviated *eofw*.

eol (str) The set of characters that indicate an end-of-line. Tip recognizes escape characters only after an end-of-line.

escape (char) The command prefix (escape) character; abbreviated *es*; default value is '~'.

exceptions (str) The set of characters that should not be discarded due to the beautification switch; abbreviated *ex*; default value is `\t\n\f\b`.

force (char) The character used to force literal data transmission; abbreviated *fo*; default value is ^P.

framesize (num) The amount of data (in bytes) to buffer between file system writes when receiving files; abbreviated *fr*.

host (str) The name of the host to which you are connected; abbreviated *ho*.

prompt (char) The character that indicates and end-of-line on the remote host; abbreviated *pr*; default value is \n. This value is used to synchronize during data transfers. The count of lines transferred during a file transfer command is based on recipt of this character.

raise (bool) Upper case mapping mode; abbreviated *ra*; default value is *off*. When this mode is enabled, all lower case letters are mapped to upper case by *tip* for transmission to the remote machine.

raisechar (char) The input character used to toggle upper case mapping mode; abbreviated *rc*; default value is ^A.

record (str) The name of the file in which a session script is recorded; abbreviated *rec*; default value is `tip.record`.

script (bool) Session scripting mode; abbreviated *sc*; default is *off*. When *script* is *true*, tip records everything transmitted by the remote machine in the script record file specified in *record*. If the *beautify* switch is on, only printable ASCII characters are included in the script file (those characters betwee 040 and 0177). The variable *exceptions* is used to indicate characters that are an exception to the normal beautification rules.

tabexpand (bool) Expand tabs to spaces during file transfers; abbreviated *tab*; default value is *false*. Each tab is expanded to eight spaces.

verbose (bool) Verbose mode; abbreviated *verb*; default is *true*. When verbose mode is enabled, *tip* displays messages while dialing, shows the current number of lines transferred during a file transfer operations, and more.

SHELL (str) The name of the shell to use for the ~! command; default value is `/bin/sh`, or taken from the environment.

HOME (str) The home directory to use for the ~c command; default value is taken from the environment.

NOTES

- The full set of variables is undocumented.

FILES

/etc/remote	global system descriptions
/etc/phones	global phone number data base
${REMOTE}	private system descriptions
${PHONES}	private phone numbers
~/.tiprc	initialization file.
/usr/spool/uucp/LCK..*	lock file to avoid conflicts with uucp

SEE ALSO

Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: remote, phones

touch

update date last modified of a file

The touch command attempts to set the modified date of each specified file. If a file exists, touch reads a character and writes it back.

---

touch [-c] [-f] ***file...***

ARGUMENTS

-c prevents file creation. By default, if *file* does not exist, touch makes an attempt to create it.

-f attempts to force the touch in spite of read and write permissions on *file*.

file specifies a file to modify.

SEE ALSO

Concentrix-2800 System Reference: System Calls and C Library Routines: utimes

tr

translate characters

The tr command copies the standard input to the standard output with substitution or deletion of selected characters.

---

tr [-cds] [*string1* [*string2*]]

ARGUMENTS

-c	complements the set of characters in *string1* with respect to the set of characters whose ASCII codes are 01 through 0377 octal.
-d	deletes all input characters in *string1*.
-s	squeezes all strings of repeated output characters that are in *string2* to single characters.
string1	maps input characters found in *string1* into the corresponding characters of *string2*.
string2	pads *string2* (if short) to the length of *string1* by duplicating its last character.

DESCRIPTION

In either string the notation *a–b* means a range of characters from *a* to *b* in increasing ASCII order. The backslash character followed by 1, 2 or 3 octal digits stands for the character whose ASCII code is given by those digits. A backslash followed by any other character stands for that character.

EXAMPLE

The following example creates a list of all the words in `file1` one per line in `file2`, where a word is taken to be a maximal string of alphabetics. The second string is quoted to protect backslash from the Shell. 012 is the ASCII code for newline.

```
tr -cs A-Za-z '\012' <file1 >file2
```

NOTES

- Tr does not handle ASCII NUL in *string1* or *string2*; it always deletes NUL from input.

SEE ALSO

ed, expand
Concentrix System User's Guide: ASCII

trap

read and execute command upon receiving signal (Bourne Shell)

The trap command sets up the signal(s) for reading and executing a command.

---

trap [*arg*] [*n*] ...

ARGUMENTS

arg is the command to be read and executed when the shell receives signal(s)*n*.

n is the signal number. Trap commands are executed in order of signal number.

NOTES

- *arg* is scanned once when the trap is set, and once when the trap is taken.
- If *arg* is absent, all trap(s) *n* are reset to their original values.
- If *arg* is a null string, the signal is ignored by the shell and by invoked commands.
- If *n* is 0, the command *arg* is executed on exit from the shell, otherwise upon receipt of signal *n* as numbered in the system call sigvec. Trap with no arguments prints a list of commands asociated with each signal number.

SEE ALSO

sigvec
Concentrix System User's Guide, Bourne Shell Reference

troff

text formatting and typesetting (unsupported)

The troff command formats text in the named files for printing on a Graphic Systems C/A/T phototypesetter.

---

troff [*option...*] [*file...*]

ARGUMENTS

The options can appear in any order so long as they appear before the files. The first seven options (–o through –q) are common to nroff and troff. The last seven options (–t through –F) are unique to troff.

–o*list* prints only pages whose page numbers appear in the comma-separated *list* of numbers and ranges. A range *N–M* means pages *N* through *M*; an initial *–N* means from the beginning to page *N*; and a final *N–* means from *N* to the end.

–n*N* numbers first generated page *N*.

–s*N* stops every *N* pages. Nroff halts prior to every *N* pages (default *N*=1) to allow paper loading or changing, and resumes upon receipt of a newline. Troff stops the phototypesetter every *N* pages, produces a trailer to allow changing cassettes, and resumes when the typesetter's start button is pressed.

–m*name* prepends the macro file /usr/lib/tmac/tmac.*name* to the input *files*.

–r*aN* sets register *a* (one-character) to *N*.

–i reads standard input after the input files are exhausted.

–q invokes the simultaneous input-output mode of the *rd* request.

–t directs output to the standard output instead of the phototypesetter.

–f refrains from feeding out paper and stopping phototypesetter at the end of the run.

–w waits until the phototypesetter is available, if currently busy.

–b reports whether the phototypesetter is busy or available. No text processing is done.

–a sends a printable ASCII approximation of the results to the standard output.

-pN prints all characters in point size *N* while retaining all prescribed spacings and motions, to reduce phototypesetter elapsed time.

-F*fontdir* produces output for devices besides the phototypesetter. The directory *fontdir* contains the font width tables, instead of the default directory /usr/lib/fonts.

file specifies an input file. A hyphen specifies the standard input (the default).

DESCRIPTION

If the file /usr/adm/tracct is writable, troff keeps phototypesetter accounting records there. The integrity of that file can be secured by making troff a "set user-id" program.

FILES

/tmp/ta*	temporary file
/usr/lib/tmac/tmac.*	standard macro files
/usr/lib/term/*	terminal driving tables for nroff
/usr/lib/font/*	font width tables for troff
/dev/cat	phototypesetter
/usr/adm/tracct	accounting statistics for /dev/cat

SEE ALSO

col, eqn, tbl
Document Processing Guide: Using Nroff/Troff

true

provide truth values

The true command returns an exit status of zero. True and false are usually used in a Bourne shell script. They test for the appropriate status "true" or "false" before running (or failing to run) a list of commands.

---

true

EXAMPLE

```
while true
do
        command-list
done
```

SEE ALSO

csh, sh, false

tset

terminal initialization

Tset sets up your terminal when you first log in to a Concentrix or Concentrix-2800 system. It does terminal dependent processing such as setting erase and kill characters, setting or resetting delays, sending any sequences needed to properly initialized the terminal, and the like.

Tset now includes the functions of the old reset command. See below for more explanation.

---

tset [*options...*] [**-m** [*ident*][*test baudrate*]:*type*]... [*type*]

reset [*options...*] [**-m** [*ident*][*test baudrate*]:*type*]... [*type*]

ARGUMENTS

Some of the following options accept a character argument *c*. Tset assumes a control character but accepts *c* in the form circumflex-letter. For example: -c ^H is the same as -c CTRL H.

-e*c* set the erase character to be the named character c on all terminals, the default being the backspace character on the terminal, usually ^H.

-k*c* is similar to -e but for the line kill character rather than the erase character; c defaults to ^X (for purely historical reasons). The kill characters is left alone if -k is not specified.

-i*c* s similar to -e but for the interrupt character rather than the erase character; c defaults to ^C.

-s Print the sequence of csh commands to initialize the environment variables TERM and TERMCAP based on the name of the terminal finally decided upon.

The name of the terminal finally decided upon is output on the standard output. This is intended to be captured by the shell and placed in the environment variable TERM.

-n On systems with the Berkeley 4BSD tty driver, specifies that the new tty driver modes should be initialized for this terminal. For a CRT, the CRTERASE and CRTKILL modes are set only if the baud rate is 1200 or greater. See tty(4) for more detail.

-I suppresses transmitting terminal initialization strings.

-Q suppresses printing the "Erase set to" and "Kill set to" messages.

DESCRIPTION

Tset sets up your terminal when you first log in to a UNIX system. It does terminal dependent processing such as setting erase and kill characters, setting or resetting delays, sending any sequences needed to properly initialized the terminal, and the like. It first determines the type of terminal involved, and then does necessary initializations and mode settings. The type of terminal attached to each port is specified in the /etc/ttys database. Type names for terminals may be found in the termcap database. If a port is not wired permanently to a specific terminal (not hardwired) it will be given an appropriate generic identifier such as dialup.

In the case where no arguments are specified, tset simply reads the terminal type out of the environment variable TERM and re-initializes the terminal. The rest of this manual concerns itself with mode and environment initialization, typically done once at login, and options used at initialization time to determine the terminal type and set up terminal modes.

When used in a startup script (.profile for sh users or .login for *csh* users) it is desirable to give information about the type of terminal you will usually use on ports which are not hardwired. These ports are identified in /etc/ttys as dialup or plugboard or arpanet, etc. To specify what terminal type you usually use on these ports, the -m (map) option flag is followed by the appropriate port type identifier, an optional baud rate specification, and the terminal type. (The effect is to "map" from some conditions to a terminal type, that is, to tell tset "If I'm on this kind of port, guess that I'm on that kind of terminal".) If more than one mapping is specified, the first applicable mapping prevails. A missing port type identifier matches all identifiers. Any of the alternate generic names given in termcap may be used for the identifier.

A baudrate is specified as with stty, and is compared with the speed of the diagnostic output (which should be the control terminal). The baud rate test may be any combination of: >, @, <, and !; @ means "at" and ! inverts the sense of the test. To avoid problems with metacharacters, it is best to place the entire argument to -m within ' (single quote) characters; users of *csh* must also put a "\" before any "!" used here.

Thus

```
tset -m 'dialup>300:adm3a' -m dialup:dw2 -m
'plugboard:?adm3a'
```

causes the terminal type to be set to an adm3a if the port in use is a dialup at a speed greater than 300 baud; to a dw2 if the port is (otherwise) a dialup (i.e. at 300 baud or less). (NOTE: the examples given here appear to take up more than one line, for text processing reasons. When you type in real tset commands, you must enter them entirely on one line.) If the type finally determined by tset begins with a question mark, the user is asked if s/he really wants that type. A null response means to use that type; otherwise, another type can be entered which will be used instead. Thus, in the above case, the user will be queried on a plugboard port as to whether they are actually using an adm3a.

If no mapping applies and a final type option, not preceded by a -m, is given on the command line then that type is used; otherwise the type found in the /etc/ttys database will be taken to be the terminal type. This should always be the case for hardwired ports.

It is usually desirable to return the terminal type, as finally determined by tset, and information about the terminal's capabilities to a shell's environment. This can be done using the option; using the Bourne shell, sh:

```
export TERM; TERM=`tset options...`
```

or using the C shell, csh:

```
setenv TERM `tset options...`
```

With csh it is preferable to use the following command in your .login file to initialize the TERM and TERMCAP environment variables at the same time.

```
eval `tset -s options...`
```

It is also convenient to make an alias in your .cshrc:

```
alias tset 'eval `tset -s \!*`'
```

This allows the command:

```
tset 2621
```

to be invoked at any time to set the terminal and environment. Note to Bourne Shell users: It is not possible to get this aliasing effect with a shell script, because shell scripts cannot set the environment of their parent. (If a process could set its parent's environment, none of this nonsense would be necessary in the first place.)

These commands cause tset to place the name of your terminal in the variable TERM in the environment; see environ(7).

Once the terminal type is known, tset engages in terminal driver mode setting. This normally involves sending an initialization sequence to the terminal, setting the single character erase (and optionally the line-kill (full line erase)) characters, and setting special character delays. Tab and newline expansion are turned off during transmission of the terminal initialization sequence.

On terminals that can backspace but not overstrike (such as a CRT), and when the erase character is the default erase character ('#' on standard systems), the erase character is changed to BACKSPACE (Control-H).

NOTES

- If tset is invoked as reset, it will set cooked and echo modes, turn off cbreak and raw modes, turn on newline translation, and restore special characters to a sensible state before any terminal dependent processing is done. Any special character that is found to be NULL or "-1" is reset to its default value. All arguments to tset may be used with reset.

  This is most useful after a program dies leaving a terminal in a funny state. You may have to type "<LF>reset<LF>" to get it to work since <CR> may not work in this state. Often none of this will echo.

- The tset command is one of the first commands a user must master when getting started on a UNIX system. Unfortunately, it is one of the most complex, largely because of the extra effort the user must go through to get the environment of the login shell set. Something needs to be done to make all this simpler,

either the login program should do this stuff, or a default shell alias should be made, or a way to set the environment of the parent should exist.

- This program can't intuit personal choices for erase, interrupt and line kill characters, so it leaves these set to the local system standards.

EXAMPLES

These examples all assume the Bourne shell and use the option. If you use *csh*, use one of the variations described above. Note that a typical use of *tset* in a .profile or .login will also use the –e and –k options, and often the –n or –Q options as well. These options have not been included here to keep the examples small. (NOTE: some of the examples given here appear to take up more than one line, for text processing reasons. When you type in real *tset* commands, you must enter them entirely on one line.)

At the moment, you are on a 2621. This is suitable for typing by hand but not for a .profile, unless you are always on a 2621.

```
export TERM; TERM=`tset 2621`
```

You have an h19 at home which you dial up on, but your office terminal is hardwired and known in /etc/ttys.

```
export TERM; TERM=`tset -m dialup:h19`
```

You have a switch which connects everything to everything, making it nearly impossible to key on what port you are coming in on. You use a vt100 in your office at 9600 baud, and dial up to switch ports at 1200 baud from home on a 2621. Sometimes you use someone elses terminal at work, so you want it to ask you to make sure what terminal type you have at high speeds, but at 1200 baud you are always on a 2621. Note the placement of the question mark, and the quotes to protect the greater than and question mark from interpretation by the shell.

```
export TERM; TERM=`tset -m 'switch>1200:?vt100' -m
'switch<=1200:2621'
```

All of the above entries will fall back on the terminal type specified in /etc/ttys if none of the conditions hold. The following entry is appropriate if you always dial up, always at the same baud rate, on many different kinds of terminals. Your most common terminal is an adm3a. It always asks you what kind of terminal you are on, defaulting to adm3a.

```
export TERM; TERM=`tset ?adm3a`
```

If the file /etc/ttys is not properly installed and you want to key entirely on the baud rate, the following can be used:

```
export TERM; TERM=`tset -m '>1200:vt100' 2621`
```

Here is a fancy example to illustrate the power of tset and to hopelessly confuse anyone who has made it this far. You dial up at 1200 baud or less on a concept100, sometimes over switch ports and sometimes over regular dialups. You use various terminals at speeds higher than 1200 over switch ports, most often the terminal in your office, which is a vt100. However, sometimes you log in from the university you used to go to, over

the ARPANET; in this case you are on an ALTO emulating a dm2500. You also often log in on various hardwired ports, such as the console, all of which are properly entered in /etc/ttys. You want your erase character set to control H, your kill character set to control U, and don't want tset to print the "Erase set to Backspace, Kill set to Control U" message.

```
export TERM; TERM=`tset -e -k^U -Q -m 'switch<=1200:concept100' -m
'switch:?vt100' -m dialup:concept100 -m arpanet:dm2500`
```

FILES

/etc/ttys	port name to terminal type mapping database
/etc/termcap	terminal capability database

SEE ALSO

csh, sh, stty
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: ttys, termcap

tsort

topological sort

The tsort command produces on the standard output a totally ordered list of items consistent with a partial ordering of items mentioned in the input *file*. The input consists of pairs of items (nonempty strings) separated by blanks. Pairs of different items indicate ordering. Pairs of identical items indicate presence, but not ordering.

---

tsort *file*

ARGUMENTS

file specifies an input file. The default is the standard input.

DIAGNOSTICS

- `Odd data`: there is an odd number of fields in the input file.

NOTES

- Tsort uses a quadratic algorithm; not worth fixing for the typical use of ordering a library archive file.

SEE ALSO

lorder

tty

get terminal name

The tty command displays the pathname of the user's terminal and returns an exit value of zero if the standard input is a terminal and one if it is not.

---

tty [-s]

ARGUMENTS

-s (silent) returns an exit value without displaying the pathname.

DIAGNOSTICS

- `not a tty` – if the standard input file is not a terminal.

ul

convert underlining for terminal

The ul command reads files and translates underscores to the sequence that indicates underlining for the terminal in use, as specified by the environment variable TERM.

---

ul [-i] [-t *terminal*] [*name...*]

ARGUMENTS

-i	indicates underlining with a separate line of hyphens (useful when you want to look at the underlining that is present in an *nroff* output stream on a video terminal).
-t *terminal*	overrides the terminal type specified in the environment.
name	specifies an input file (the default is the standard input).

DESCRIPTION

The file /etc/termcap is read to determine the appropriate sequences for underlining. If the terminal is incapable of underlining, but is capable of a standout mode, that is used instead. If the terminal can overstrike, or handles underlining automatically, ul degenerates to cat. If the terminal cannot underline, underlining is ignored.

NOTES

- Nroff usually outputs a series of backspaces and underlines intermixed with the text to indicate underlining. No attempt is made to optimize the backward motion.

SEE ALSO

man, nroff, colcrt

umask

set file creation mask (C shell)

The umask command sets your file creation mask to a specified value. The file creation mask specifies the access codes given to a file when it is created.

---

umask [*value*]

ARGUMENTS

value is an octal number that represents a 12-bit binary file creation mask as described below. If you omit *value*, the current mask is displayed.

DESCRIPTION

The file creation mask is the *logical complement* of the desired file access code (permission) as shown below. For example, if you want a file access code of 755 (all normal permissions except write access for group or others), you would use a creation mask of 022.

bit	who	permission
0	others	execute
1	others	write
2	others	read
3	group	execute
4	group	write
5	group	read
6	owner	execute
7	owner	write
8	owner	read
9	sticky (see the system call *chmod*)	
10	set group ID on execution	
11	set user ID on execution	

NOTES

- When you set up the file creation mask so that it gives execute permission to someone, that permission is only applied to *directories*. No *file* that you create will be given execute permission. Execute permission for a file can only be granted with the chmod command.

SEE ALSO

chmod

umask

set file creation mask (Bourne shell)

The umask command sets your file creation mask to a specified octal value *nnn*. The file creation mask specifies the access codes given to a file when it is created.

---

umask [*value*]

ARGUMENTS

value is an octal number that represents a 12-bit binary file creation mask as described below. If you omit *value*, the current mask is displayed.

DESCRIPTION

The file creation mask is the *logical complement* of the desired file access code (permission) as shown below. For example, if you want a file access code of 755 (all normal permissions except write access for group or others), you would use a creation mask of 022.

bit	who	permission
0	others	execute
1	others	write
2	others	read
3	group	execute
4	group	write
5	group	read
6	owner	execute
7	owner	write
8	owner	read
9	sticky (see the system call *chmod*)	
10	set group ID on execution	
11	set user ID on execution	

NOTES

- When you set up the file creation mask so that it gives execute permission to someone, that permission is only applied to *directories*. No *file* that you create will be given execute permission. Execute permission for a file can only be granted with the chmod command.

SEE ALSO

chmod

unalias

remove alias (C shell)

The unalias command removes an alias (a name bound to a command string by the alias command).

---

unalias *pattern*

ARGUMENTS

pattern specifies which aliases to remove.

EXAMPLE

The following command removes all aliases.

```
unalias *
```

NOTES

- It is not an error if no aliases are removed.

SEE ALSO

alias

uncompact

expand files that were compressed with the compact utility (old command, found in /usr/old, replaced by uncompress)

The uncompact command restores a file compressed by the compact command. The standard output is used by default.

---

uncompact [*name...*]

ARGUMENTS

name specifies an input file. The default is the standard input.

DESCRIPTION

Compact operates as an on-line algorithm. Each time a byte is read, it is encoded immediately according to the current prefix code. This code is an optimal Huffman code for the set of frequencies seen so far. It is unnecessary to prepend a decoding tree to the compressed file since the encoder and the decoder start in the same state and stay synchronized. Furthermore, compact and uncompact can operate as filters.

When an argument *file* is given, it is compacted and the resulting file is placed in *file*.C; *file* is unlinked. The first two bytes of the compacted file code the fact that the file is compacted. This code is used to prohibit recompaction.

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

NOTES

- The last segment of the filename must contain fewer than thirteen characters to allow space for the appended .C.

FILES

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

SEE ALSO

compact, ccat,
Variations on a Theme of Huffman, Gallager, Robert G.
I.E.E.E. Transactions on Information Theory,
vol. IT-24, no. 6, November 1978, pp. 668 - 674.

uncompress

expand files that were compressed with the compress utility

The uncompress command expands files that were compressed using the adaptive Lempel–Ziv coding of the compress utility.

---

uncompress [–f] [–v] [–c] [*name...*]

ARGUMENTS

name the input file that you want to expand. The input file must have been compressed by the compress utility, and must have a .Z extension. The resulting expanded file is given the same name as the input file, but without the .Z extension. If no input filename is given, the standard input is expanded to the standard output.

–f forces the expansion of *name*, even if the output file (without the .Z extension) already exists. Except when run in the background under /bin/sh or if –f is given, the user is prompted as to whether an existing output file should be overwritten.

–v causes uncompress to print a message as it expands each input file.

–c makes uncompress write to the standard output, without changing any files (the "cat" option).

DESCRIPTION

Uncompress expands files that have been compressed with the adaptive Lempel–Ziv coding of the compress utility.

Every file that is expanded must have a .Z extension. If a filename that you supply does not end in .Z, uncompress appends a .Z to the name you supply, before looking for the file.

After creating an expanded version of a file, uncompress deletes the compressed version.

Uncompress –c allows you to look at the expanded version of a file, without altering the compressed file. The zcat command also performs the same function.

DIAGNOSTICS

- Usage: compress [–fvc] [file ...]. You specified invalid options on the command line.
- *name* already exists; do you wish to overwrite (y or n)? Respond "y" if you want the output file to be replaced; "n" if not.

- *name*: not in compressed format. The file specified to uncompress has not been compressed.
- *name*: compressed with xx bits, can only handle yy bits. File was compressed by a program that could deal with more bits than the compress code on this machine. Recompress the file with smaller bits.
- uncompress: corrupt input. A SIGSEGV violation was detected which usually means that the input file is corrupted.
- *name* -- has xx other links: unchanged. The input file has hard links; it is left unchanged. See ln for more information about links.

NOTES

- Uncompress should be more flexible about the existence (or nonexistence) of the '.Z' suffix on compressed files.
- Uncompress does not work well with links. If a file has a hard link to it, *uncompress* will not expand the file, regardless of whether you use the file name or the link name.

FILES

*.Z	compressed file created by compress, removed by uncompress.

SEE ALSO

compress, zcat

Variations on a Theme of Huffman, Gallager, Robert G.,
I.E.E.E. Transactions on Information Theory, vol. IT-24, no. 6,
November 1978, pp. 668 - 674.

unexpand

compress spaces to tabs

The unexpand command processes the named files changing spaces to tabs. By default, only leading spaces and tabs are converted to maximal strings of tabs.

---

unexpand [-a] [*file...*]

ARGUMENTS

-a inserts tabs whenever they would compress the resultant file by replacing two or more characters.

file specifies an input file.

SEE ALSO

expand

unhash

disable hash table (C shell)

The unhash command disables the use of the internal hash table that speeds location of executed programs.

---

unhash

SEE ALSO

hashstat, rehash

unifdef

remove ifdef lines from a C program

Unifdef is useful for removing chosen ifdef lines from a C program while otherwise leaving the file alone.

---

unifdef [-t -l -c -D*sym* -U*sym* -id*sym* -iu*sym*]... [*file*]

ARGUMENTS

-t	Input is plain text (not C code). This option makes unifdef refrain from attempting to recognize comments and single and double quotes.
-l	Replace removed lines with blank lines instead of deleting them.
-c	The operation of unifdef is complemented, i.e. the lines that would have been removed or blanked are retained and vice versa.
-D*sym*	Make this symbol defined as the C program is scanned.
-U*sym*	Make this symbol undefined as the C program is scanned.
-id*sym*	Make this symbol defined, but don't parse (ignore) the content of the lines inside these ifdef blocks.
-iu*sym*	Make this symbol undefined, but don't parse (ignore) the content of the lines inside these ifdef blocks.

DESCRIPTION

Unifdef is useful for removing ifdef lines from a C source file while otherwise leaving the file alone. Unifdef is like a stripped-down C preprocessor: it is smart enough to deal with the nested ifdefs, comments, single and double quotes of C syntax so that it can do its job, but it doesn't do any including or interpretation of macros.

Unifdef copies its output to stdout. If no input file is specified, unifdef will take its input from stdin.

You specify which symbols you want defined (with -D) or undefined (with -U) and the lines inside those ifdefs will be copied to the output or removed as appropriate. The ifdef, ifndef, else, and endif lines associated with *sym* will also be removed. Ifdefs involving symbols you don't specify are untouched and copied out along with their associated ifdef, else, and endif lines.

Unifdef does not strip out comments, though it recognizes and ignores them.

If an "ifdef X" occurs nested inside another "ifdef X", then the inside ifdef is treated as if it were an unrecognized symbol. If the same symbol appears in more than one argument, only the first occurrence is significant.

If you use ifdefs to delimit non-C lines, such as comments or code that is under construction, then you must tell unifdef which symbols are used for that purpose. You do this with the **-id** and **-iu** options. Unifdef will still copies these ifdef lines to output, but it won't try to parse the lines for quotes and comments.

If you want to use unifdef for plain text (not C code), use the **-t** option. This makes unifdef refrain from attempting to recognize comments, and single and double quotes.

If the **-c** argument is specified, then the operation of unifdef is complemented, i.e. the lines that would have been removed or blanked are retained, and vice versa.

DIAGNOSTICS

Exit status is 0 if output is exact copy of input, 1 if not, 2 if trouble.

- Premature EOF, inappropriate else or endif.

NOTES

- Does not know how to deal with cpp constructs such as:

```
#if defined(X) || defined(Y)
```

SEE ALSO

diff

uniq

report repeated lines in a file

The uniq command reads the input file comparing adjacent lines. In the normal case, the second and succeeding copies of repeated lines are removed; the remainder is written on the output file. Repeated lines must be adjacent in order to be found (see sort).

---

uniq [-udc [+*n*] [-*n*]] [*input* [*output*]]

ARGUMENTS

-u outputs only the lines that are not repeated in the original file.

-d writes one copy of only the repeated lines.

-c supersedes -u and -d and generates an output report in default style but with each line preceded by a count of the number of times it occurred.

The *n* arguments specify skipping an initial portion of each line in the comparison. A field is defined as a string of non-space, non-tab characters separated by tabs and spaces from its neighbors.

+*n* ignores the first *n* characters. Fields are skipped before characters.

-*n* ignores the first *n* fields together with any blanks before each.

SEE ALSO

sort, comm

units

quantitative conversion program

The units command converts quantities expressed in various standard scales to their equivalents in other scales.

---

units

DESCRIPTION

Units works interactively in this fashion:

```
You have: inch
You want: cm
        * 2.54000e+00
        / 3.93701e-01
```

A quantity is specified as a multiplicative combination of units optionally preceded by a numeric multiplier. Powers are indicated by suffixed positive integers, division by the usual sign:

```
You have: 15 pounds force/in2
You want: atm
        * 1.02069e+00
        / 9.79730e-01
```

Units only does multiplicative scale changes. Thus it can convert Kelvin to Rankine, but not Centigrade to Fahrenheit. Most familiar units, abbreviations, and metric prefixes are recognized, together with a generous leavening of exotica and a few constants of nature including:

pi	ratio of circumference to diameter
c	speed of light
e	charge on an electron
g	acceleration of gravity
force	same as g
mole	Avogadro's number
water	pressure head per unit height of water
au	astronomical unit

"Pound" is a unit of mass. Compound names are run together, e.g. "lightyear". British units that differ from their US counterparts are prefixed thus: "brgallon". Currency is denoted "belgiumfranc", "britainpound", ...

For a complete list of units, examine `/usr/lib/units`.

FILES

/usr/lib/units

unlimit

remove resource limitation (C shell)

The unlimit command removes the limitations on the specified system resource.

---

unlimit [-h] [*resource*]

ARGUMENTS

-h remove the specified hard limits. Only the super-user may do this.

resource is one of:

`cputime` the maximum number of cpu-seconds to be used by each process.

`filesize` the largest single file that can be created.

`datasize` the maximum growth of the data+stack region via *sbrk (System Calls)* beyond the end of the program text.

`stacksize`
the maximum size of the automatically-extended stack region.

`coredumpsize`
the size of the largest core dump that will be created.

DESCRIPTION

When a regular user (not a superuser) unlimits a resource, the limit for that resource becomes the hard limit. That is, the hard limit is an upper limit that a user may not exceed.

If *resource* is omitted, all limitations are removed.

SEE ALSO

limit

unset

remove local variable (C shell)

The unset command removes user-defined variables that are local to the current C shell.

---

unset *pattern*

ARGUMENTS

pattern specifies the variable to remove.

NOTES

- Do not `unset *`. That removes all variables, including shell variables, and can cause undesirable effects.
- It is not an error if no variables are removed.

SEE ALSO

set, setenv, unsetenv

unsetenv

remove environment variable (C shell)

The unsetenv command removes an environment variable (see setenv).

---

unsetenv *pattern*

ARGUMENTS

pattern specifies the variable to remove.

NOTES

- Do not `unsetenv *`. That removes all environment variables, including shell variables, and can cause undesirable effects.
- It is not an error if no environment variables are removed.

SEE ALSO

set, setenv, unset

uptime

show how long system has been up

The uptime command displays the current time; the length of time the system has been up; the current number of users; and the average number of jobs in the run queue over the last 1, 5, 15, and 60 minutes. The output of this command is the same as the first line of a *w* command.

---

uptime

FILES

/vmunix	system name list

SEE ALSO

w

users

display list of current system users

The users command lists the login names of the users currently on the system in a compact, one-line format.

---

users

NOTES

- The Concentrix command u is a synonym for users and can be used in its place.

FILES

/etc/utmp

SEE ALSO

who

uucp

unix to unix copy

The uucp command copies files from one UNIX system to another.

---

uucp [-acCdfmr] [-n*user*] [-g*grade*] [-s*spool*] [-x*debug*] *source....* *destination*

ARGUMENTS

-a	avoids doing a getwd to find the current directory. (This is sometimes used for efficiency.)
-c	uses the source file when copying out rather than copying the file to the spool directory. (This is the default.)
-C	copies the source file to the spool directory and transmits the copy.
-d	makes all necessary directories for the file copy. (This is the default.)
-f	does not make intermediate directories for the file copy.
-m	sends mail to the requester when the copy is complete.
-r	does not start the transfer, just queues the job.
-n*user*	notifies *user* on remote system (i.e., send *user* mail) that a file was sent.
-g*grade*	controls when a job is transmitted during a particular conversation. *Grade* is a single letter/number; lower ASCII sequence characters will cause a job to be transmitted earlier. Default is 'n'. By way of comparison, *uux* defaults to 'A'; *mail* is usually sent at 'C'.
-s*spool*	uses *spool* as the spool directory instead of the default.
-x*debug*	turns on the debugging at level *debug*.
source...	specifies a file to copy.
destination	specifies where to copy the file.

DESCRIPTION

Uucp copies files named by the *source* arguments to the *destination* argument. A file name may be a pathname on your machine, or may have the form

system–name!*pathname*

where *system–name* is taken from a list of system names that *uucp* knows about. Shell metacharacters ?*[] appearing in the pathname part will be expanded on the appropriate system.

From the C shell, the exclamation (!) must be escaped (\!) to inhibit the history mechanism.

Pathnames may be one of:

- a full pathname;
- a pathname preceded by *-user*; where *user* is a userid on the specified system and is replaced by that user's login directory;
- a pathname prefixed by ~, where ~ is expanded into the system's public directory (usually /usr/spool/uucppublic);
- a partial pathname, which is prefixed by the current directory.

If the result is an erroneous pathname for the remote system, the copy will fail. If the *destination* is a directory, the last part of the *source* name is used.

Uucp preserves execute permissions across the transmission and gives 0666 read and write permissions (see chmod).

NOTES

- All files received by uucp will be owned by the uucp administrator (usually UID 5).
- The **–m** option will only work sending files or receiving a single file. (Receiving multiple files specified by special shell characters ?*[] will not activate the **–m** option.)
- At present uucp cannot copy to a system several "hops" away, that is, a command of the form

  uucp myfile system1!system2!system3!yourfile

  is not permitted. Use uusend instead.
- Uucp refuses to copy a file that does not give read access to "other;" that is, the file must have at least 0444 modes.
- The domain of remotely accessible files can (and for obvious security reasons, usually should) be severely restricted. You will very likely not be able to fetch files by pathname; ask a responsible person on the remote system to send them to you. For the same reasons you will probably not be able to send files to arbitrary pathnames.

FILES

/usr/spool/uucp	spool directory
/usr/lib/uucp/*	other data and program files

SEE ALSO

uux, mail
Concentrix-2800 System Administration: UUCP Dial-Up Network, UUCP Administrator's Manual

uudecode

decode binary file from mail

The uudecode command reads an encoded file, strips off any leading and trailing lines added by mailers, and recreates the original file with the specified mode and name. It is used with uuencode to send binary files via uucp (or other) mail. This combination can be used over indirect mail links even when uusend is not available.

---

uudecode [*file*]

ARGUMENTS

file specifies an encoded file.

NOTES

- If these facilities are not available, the file can be sent to a user on the remote machine who can uudecode it manually.
- The file is expanded by 35 per cent (three bytes become four plus control information) causing it to take longer to transmit.
- The user on the remote system who is invoking uudecode (often uucp) must have write permission on the specified file.

SEE ALSO

uusend, uucp, uux, mail
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: uuencode
Concentrix-2800 System Administration: UUCP Dial-Up Network, UUCP Administrator's Manual

uuencode

encode binary file for mail

The uuencode command takes the named source file and produces an encoded version on the standard output. It is used with uudecode to send binary files via uucp (or other) mail. This combination can be used over indirect mail links even when uusend is not available.

---

uuencode [*source*] *remotedest*

ARGUMENTS

source specifies a source file (the default is the standard input).

remotedest specifies a remote destination file.

DESCRIPTION

The encoding uses only printing ASCII characters, and includes the mode of the file and the *remotedest* for recreation on the remote system.

The intent is that all mail to the user decode should be filtered through the uudecode program. This way the file is created automatically without human intervention. This is possible on the uucp network by either using sendmail or by making rmail be a link to Mail instead of mail. In each case, an alias must be created in a master file to get the automatic invocation of uudecode.

The encode file has an ordinary text form and can be edited by any text editor to change the mode or remote name.

NOTES

- The file is expanded by 35 per cent (three bytes become four plus control information) causing it to take longer to transmit.
- The user on the remote system who is invoking uudecode (often uucp) must have write permission on the specified file.

SEE ALSO

atob, uusend, uucp, uux, mail
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: uuencode
Concentrix-2800 System Administration: UUCP Dial-Up Network, UUCP Administrator's Manual

uulog

display UUCP log files

Queries a log of uucp and uux transactions in the file /usr/spool/uucp/LOGFILE and prints information as specified by the options.

---

uulog [*option* ...]

ARGUMENTS

-s*sys* prints information about work involving the specified system.

-u*user* prints information about work involving the specified user.

FILES

/usr/spool/uucp/LOGFILE

SEE ALSO

uucp, uux

NOTES

Early releases of the UUCP facility used uulog to merge separate, partial log files. This capability is no longer needed and is no longer supported.

BUGS

The recording of which users issued requests is unreliable.

uuname

list names of UUCP hosts

The uuname command lists the names of all known systems that can be reached via UUCP (all those in the file /usr/lib/uucp/L.sys).

---

uuname [-l]

ARGUMENTS

-l returns the local system name; this may differ from the *hostname* for the system if the *hostname* is very long.

SEE ALSO

hostname, uucp, uux
Concentrix-2800 System Administration: UUCP Dial-Up Network, UUCP Administrator's Manual

uuq

examine or manipulate the uucp queue

The uuq command examines (and possibly deletes) entries in the UUCP queue.

---

uuq [-l] [-h] [-s*system*] [-u*user*] [-d*jobno*] [-r*sdir*] [-b*baud*]

ARGUMENTS

-h prints only the summary lines for each system. Summary lines give system name, number of jobs for the system, and total number of bytes to send.

-l specifies a long format listing. The default is to list only the job numbers sorted across the page.

-s*system* limits output to jobs for systems whose system names begin with system.

-u*user* limits output to jobs for users whose login names begin with user.

-d*jobno* deletes job number jobno (as obtained from a previous uuq command) from the UUCP queue. Only the UUCP administrator is permitted to delete jobs.

-r*sdir* looks for files in the spooling directory *sdir* instead of the default directory.

-b*baud* uses *baud* to compute the transfer time instead of the default 1200 baud.

DESCRIPTION

When listing jobs, uuq uses a format similar to ls. For the long format, information for each job listed includes job number, number of files to transfer, user who spooled the job, number of bytes to send, type of command requested (S for ending files, R for receiving files, X for remote uucp), and file or command desired.

FILES

/usr/spool/uucp/	Default spool directory
/usr/spool/uucp/C./C.*	Control files
/usr/spool/uucp/Dhostname./D.*	Outgoing data files
/usr/spool/uucp/X./X.*	Outgoing execution files

SEE ALSO

uucp, uux, uulog
Administration Commands, uusnap

uusend

send file to remote host

The uusend command sends a file to a given location on a remote system. The system need not be directly connected to the local system, but a chain of *uucp* links must connect the two systems.

---

uusend [**-m** *mode*] *source sys1*!*sys2*!...!*destination*

ARGUMENTS

The first two options are primarily intended for internal use of *uusend*.

-m *mode* is an octal number that specifies the mode of the file on the remote end. The default is the mode of the input file.

source specifies a source file. A hyphen indicates the standard input.

destination specifies where to send the file. It can include the -userid syntax.

DIAGNOSTICS

- If anything goes wrong any further away than the first system down the line, you will never hear about it.

NOTES

- All systems along the line must have the *uusend* command available and allow remote execution of it.
- Some *uucp* systems have a bug where binary files cannot be the input to a *uux* command. If this bug exists in any system along the line, the file shows up severly munged.

SEE ALSO

uux, uucp, uuencode
Concentrix-2800 System Administration: UUCP Dial-Up Network, UUCP Administrator's Manual

uux

unix to unix command execution

The uux command gathers zero or more files from various systems, executes a command on a specified system, and sends standard output to a file on a specified system.

---

uux [-] [-cClLnprz] [-a*name*] [-g*grade*] [-x*debug*] *command*

ARGUMENTS

- **-** causes the standard input to the *uux* command to be the standard input to the *command*.

-c does not copy local file to the spool directory for transfer to the remote machine (this is the default).

-C copies the source file to the spool directory and transmits the copy.

-l tries to make a link from the original file to the spool directory; if the link fails, copies the file.

-L starts the administration command uucico with the -L option to force calls to be made to local sites only.

-n does not notify the user when the command completes (by default, the user is notified).

-p same as - (the command comes from the standard input).

-r does not start the transfer, just queues the job.

-z notifies the user only if the command fails.

-a*name* uses *name* as the user identification instead of the initiating user's identification.

-g*grade* where *grade* is a single character. A lower character in the ASCII sequence causes a job to be transmitted earlier during a particular conversation. Defaults to n. By way of comparison, *uux* defaults to A and *mail* is usually sent at C. Higher characters should be specified for high-volume jobs such as *news*.

-x*debug* turns on debugging at level *debug*, an integer in the range 0–9, where higher values provide more detailed information. The debugging output is on standard output. Debugging is permitted only if you have read access to the special file L.sys.

command is made up of one or more arguments that look like a shell command line, except that the command and file names can be prefixed by *system-name*!.

DESCRIPTION

Uux will attempt to get all files to the execution system.

Uux will notify you by mail if the requested command on the remote system was disallowed. This notification can be turned off by the -n option.

A null *system-name* is interpreted as the local system.

For files that are output files, the file name must be escaped using parentheses.

When using shell metacharacters such as (< >) angle brackets, (;) semicolon, or (|) vertical bar, quote them as individual arguments or quote the entire command.

From the C shell, the exclamation (!) must be escaped as backslash-exclamation (\!) to inhibit the history mechanism.

File names can be one of

- a full pathname.
- a pathname preceded by *~user*; where *user* is a userid on the specified system and is replaced by that user's login directory.
- a pathname prefixed by ~, where ~ expands into the system's public directory (usually /usr/spool/uucppublic).
- a partial pathname, which is prefixed by the current directory.

EXAMPLE

The command

```
uux a!wc b!/usr/file1 \(c!/usr/file2 \)
```

gets /usr/file1 from system "b" and sends it to system "a", performs a wc command on that file and sends the result of the wc command to system "c".

The command

```
uux "!diff usg!/usr/dan/file1 pwba!/a4/dan/file2
> !~/dan/file.diff"
```

will get file1 and file2 from the "usg" and "pwba" machines, execute a diff command and put the results in file.diff in the local /usr/spool/uucppublic/dan/ directory.

NOTES

- An installation can, and for security reasons generally does, limit the list of commands executable on behalf of an incoming request from uux. Typically, a restricted site permits little other than the receipt of mail via uux.
- Only the first command of a shell pipeline can have a *system-name*!. All other commands are executed on the system of the first command.

- The shell metacharacter * (asterisk) probably does not do what you want it to do.
- The shell tokens double left angle bracket (<<) and double right angle bracket (>>) are not implemented.
- There is no notification of denial of execution on the remote machine.

FILES

/usr/spool/uucp	spool directory
/usr/lib/uucp/*	other data and programs

SEE ALSO

uucp
Concentrix-2800 System Administation: UUCP Dial-Up Network, UUCP Administrator's Manual

vacation

return an "I am on vacation" indication

Vacation returns a message to people who send you mail, telling them that you are on vacation.

---

vacation -I

vacation [-j] [-a *alias*][-t *N*] *user*

ARGUMENTS

-I Initialize the vacation control files.

If the -I flag is not specified, and a *user* argument is given, vacation reads the first line from the standard input (for a from: line, no colon). If absent, it produces an error message. The following options may be specified:

-a *alias* Indicate that *alias* is one of the valid aliases for *user*, so that mail sent to that alias generates a reply.

-j Don't check whether the recipient appears in the To: or the Cc: line.

-t *N* Change the interval between repeat replies to the same sender. The default is one week. A trailing **s**, **m**, **h**, **d**, or **w** scales *N* to seconds, minutes, hours, days, or weeks, respectively.

user Directs vacation to send mail saying that *user* is on vacation.

DESCRIPTION

Vacation returns a message to people who send you mail, telling them that you are on vacation. The intended use is in a .forward file.

For example, your .forward file might contain:

```
\chuck, "|vacation eric".
```

When you receive mail, two things will happen: 1) the mail message will be forwarded to Chuck, and 2) a message will be sent to the person who sent you mail, telling them that you are on vacation.

Vacation expects to find a file, .vacation.msg, in your home directory. This file must contain a message that can be sent to each person who sends you mail. It should be an entire message (including headers). For example:

```
From: eric@ucbmonet.Berkeley.EDU (Eric Allman)
Subject: I am on vacation
Delivered-By-The-Graces-Of: the Vacation program

I am on vacation until July 22.
```

```
I will read your mail regarding "$SUBJECT" when I return.

If you have something urgent, please contact Joe Kalash at:
<kalash@ucbingres.Berkeley.EDU>.

--eric
```

This message will only be sent once a week to each different person who sends you mail.

The .vacation.msg file should include a header with at least a "Subject:" line (it should not contain a "To:" line and need not contain a "From:" line, since these are generated automatically.

If the string $SUBJECT appears in the vacations.msg file, it is replaced with the subject of the original message when the reply is sent.

The people who have sent you messages are kept in the vacation control files .vacation.pag and .vacation.dir (in your home directory). The **-I** option initializes these files, and should be executed before you modify your .forward file.

Vacation does not send a message back if the "To:" or the "Cc: "line does not list the user to whom the original message was sent or one of a number of aliases for them, if the initial "From:" line includes the string "-REQUEST@," or if a "Precedence: bulk" or "Precedence: junk" line is included in the header.

If vacation is run with no arguments, it permits you to interactively turn vacation on or off. It creats a .vacation.msg file for you, or edits an existing one, using the editor specified by the VISUAL or EDITOR environment variable, or vi if neither environment variable is set. If a .forward file is present in your home directory, it creates it for you, and automatically performs a vacation -I function, turning on vacation.

FILES

.forward	
.vacation.msg	file containing message to send to people who send you mail
.vacation.pag	vacation control file
.vacation.dir	vacation control file

SEE ALSO

Concentrix-2800 System Administration: sendmail

val

validate SCCS file

The val command determines if the specified file is an SCCS file meeting the characteristics specified by the optional argument list.

---

val -

val [-s] [-r*SID*] [-m*name*] [-y*type*] *file...*

ARGUMENTS

Arguments to val, which can appear in any order, consist of keyletter arguments beginning with a hyphen and named files. The effects of any keyletter argument apply independently to each named file on the command line.

-	is a special argument that causes reading of the standard input until an end-of-file condition is detected. Each line read is independently processed as if it were a command line argument list.
-s	silences the diagnostic message normally generated on the standard output for any error that is detected while processing each named file on a given command line.
-r*SID*	checks to determine if the specified *SID* is ambiguous (for example, r1 is ambiguous because it physically does not exist but implies 1.1, 1.2, etc. which can exist) or invalid (e. g., r1.0 or r1.1.0 are invalid because neither case can exist as a valid delta number). If the *SID* is valid and not ambiguous, a check is made to determine if it actually exists.
-m*name*	compares the argument value *name* with the SCCS %M% keyword in *file*.
-y*type*	compares the argument value *type* with the SCCS %Y% keyword in *file*.
files	specifies the files to be processed.

DESCRIPTION

Val generates diagnostic messages on the standard output for each command line and file processed and also returns a single eight-bit code upon exit that is a disjunction of the possible errors, that is, can be interpreted as a bit string where set bits are interpreted (from left to right) as follows:

bit 0 = missing file argument;
bit 1 = unknown or duplicate keyletter argument;
bit 2 = corrupted SCCS file;
bit 3 = can't open file or file not SCCS;

bit 4 = *SID* is invalid or ambiguous;
bit 5 = *SID* does not exist;
bit 6 = %Y%, -y mismatch;
bit 7 = %M%, -m mismatch;

Val can process two or more files on a given command line and in turn can process multiple command lines (when reading the standard input). In these cases an aggregate code is returned: a logical OR of the codes generated for each command line and file processed.

DIAGNOSTICS

- Use help for explanations.

NOTES

- Val can process up to 50 files on a single command line. Any number above 50 produces a core dump.

SEE ALSO

sccs, admin, delta, get, prs
Concentrix-2800 Programming Tools Guide: Source Code Control System – System V Interface.

vi

screen oriented (visual) display editor

The vi command invokes a display oriented text editor based on ex. Ex and vi run the same code; you can get to the command mode of ex from within vi and vice-versa.

---

vi [-t *tag*] [-r] [+*command*] [-l] [-w*n*] *name...*

ARGUMENTS

-t *tag* is equivalent to an initial tag command, editing the file containing the tag and positioning the cursor at its definition.

-r recovers after an editor or a system crash. It retrieves the last saved version of the named file or, if no file is specified, displays a list of saved files.

+*command* begins by executing the specified command. The default is dollar sign, positioning the editor at the last line of the first file.

-l sets up for editing Lisp, setting the *showmatch* and *lisp* options.

-w*n* sets the default window size to *n* (useful on dialups to start with a small window).

name specifies a file to edit.

NOTES

- Software tabs using CTRL T work only immediately after the *autoindent*.
- Left and right shifts on intelligent terminals do not make use of insert and delete character operations in the terminal.
- The *wrapmargin* option can be fooled since it looks at output columns when blanks are typed. If a long word passes through the margin and onto the next line without a break, the line won't be broken.
- Insert/delete within a line can be slow if tabs are present on intelligent terminals, since the terminals need help in doing this correctly.
- Saving text on deletes in the named buffers is somewhat inefficient.
- The source command does not work when executed as `:source`; there is no way to use the `:append`, `:change`, and `:insert` commands, since it is not possible to give more than one line of input to a colon escape. To use these on a `:global` you must Q to ex command mode, execute them, and then reenter the screen editor with vi or open.

FILES

/usr/lib/ex?.?strings	error messages
/usr/lib/ex?.?recover	recover command
/usr/lib/ex?.?preserve	preserve command
/etc/termcap	describes capabilities of terminals
~/.exrc	editor startup file
/tmp/Ex*nnnnn*	editor temporary
/tmp/Rx*nnnnn*	named buffer temporary
/usr/preserve	preservation directory

SEE ALSO

ex, edit,
Concentrix-2800 System User's Guide: Display Editing (vi)

vmstat

report virtual memory statistics

The vmstat command reports statistics about process, virtual memory, trap and cpu activity.

---

vmstat

EXAMPLE

```
_freelist+0x8:  13269           /* free list size */
_cnt+0xb4:      0               /* number of sleeps due to empty free list */

_cnt:           772162          /* context switches */
_cnt+0x8:       1194785         /* calls to syscall() */
_cnt+0xc:       3640568         /* device interrupts */
_cnt+0x19c:     0               /* idle switches */
_cnt+0x1a0:     16778           /* asynchronous system traps */

_cnt+0x260:     0               /* swaps */
_cnt+0x18:      0               /* pages swapped out */

_cnt+0x40:      1955            /* forks executed */
_cnt+0x88:      1438            /* exec system calls */

_cnt+0x1c:      0               /* pageins */
_cnt+0x20:      1925            /* pages paged in for private page */
_cnt+0xbc:      1000            /* page read faults for global pages */

_cnt+0x24:      0               /* pageouts */
_cnt+0x28:      0               /* pages paged out */

_cnt+0x34:      0               /* total private pages from free list */
_cnt+0x38:      14778           /* total global pages from free list */
_cnt+0x3c:      52270           /* page faults for zero fill on demand */
_cnt+0xa0:      65289           /* number of pages requested pre-zeroed */
_cnt+0xa4:      98052           /* number of deleted contents pages freed */
_cnt+0xa8:      17796           /* number of clean pages to cache */
_cnt+0xe4:      0               /* number of dirty pages passed over */
_cnt+0xe8:      0               /* number of clean pages tested */
_cnt+0xec:      0               /* number of pages reclaimed by shrinker */
```

FILES

/dev/kmem

SEE ALSO

mon

w

show who is on and what they are doing

The w command displays a summary of the current activity on the system, including what each user is doing.

---

w [-h] [-s] [-l] [*user*]

ARGUMENTS

-h suppresses the heading.

-s displays a short form of output (the login time and cpu times are left off).

-l displays the long output (the default).

user restricts the output to the specified user.

DESCRIPTION

The heading line shows the current time of day; how long the system has been up; the number of users logged into the system; and the average number of runnable jobs over the last 1, 5, 15, and 60 minutes. The fields output are: the user's login name, the name of the tty the user is on, the time of day the user logged on, the number of minutes since the user last typed anything, the CPU time used by all processes and their children on that terminal, the CPU time used by the currently active processes, the name and arguments of the current process.

NOTES

- The notion of the "current process" is muddy. The current algorithm is "the highest numbered process on the terminal that is not ignoring interrupts, or, if there is none, the highest numbered process on the terminal." This fails, for example, in critical sections of programs like the shell and editor, or when faulty programs running in the background fork and fail to ignore interrupts. (In cases where no process can be found, w displays a hyphen.)
- The CPU time is only an estimate, in particular, if someone leaves a background process running after logging out, the person currently on that terminal is "charged" with the time.
- Background processes are not shown, even though they account for much of the load on the system.
- Sometimes processes, typically those in the background, are displayed with null or garbaged arguments. In these cases, the name of the command is displayed in parentheses.

- w does not know about the new conventions for detection of background jobs. It sometimes finds a background job instead of the right one.

FILES

/etc/utmp
/dev/kmem

SEE ALSO

who, finger, ps

wait

wait for background jobs (C shell)

The wait command causes the shell to wait for all background jobs to finish. It is intended for use in shell scripts. If the shell is interactive, however, an interrupt can disrupt the wait, at which time the shell prints the names and job numbers of all jobs known to be outstanding.

---

wait

SEE ALSO

wait (Bourne shell)

wait

await completion of process (Bourne shell)

The wait command waits for a specified process and reports its termination status. The return code is that of the process waited for.

---

wait [*process*]

ARGUMENTS

process is a process identification number (pid); if not specified, all currently-active child processes are waited for.

NOTES

- Because the wait system call must be executed in the parent process, the shell itself executes wait, without creating a new process.
- Not all the processes of a three- or more-stage pipeline are children of the shell, and thus cannot be waited for.

SEE ALSO

sh, wait (C shell)

wall

write to all users

The wall command reads the standard input and sends it, preceded by `Broadcast Message ...`, to all logged in users. The sender should be super-user to override any protections the users may have invoked.

---

wall

DIAGNOSTICS

- `Cannot send to ...` when the open on a user's tty file fails.

FILES

/dev/tty?
/etc/utmp

SEE ALSO

mesg, write

WC

count words, lines, and characters

The wc command counts lines, words and characters in the named files. A word is a maximal string of characters delimited by spaces, tabs or newlines.

---

wc [-lwc] [*name...*]

ARGUMENTS

-l counts lines.

-w counts words.

-c counts characters.

name specifies an input file. The default is the standard input.

what

identify SCCS files

What searches the given files for all occurrences of the pattern that get substitutes for %Z% (this is @(#) at this printing) and prints out what follows until the first double quote, right angle-bracket, new-line, null character, or end-of-file.

---

what [-s] ***file...***

ARGUMENTS

-s quits afters finding the first occurrence of the pattern in each file.

DESCRIPTION

What is intended to be used in conjunction with the SCCS command get, which automatically inserts identifying information, but it can also be used where the information is inserted manually.

EXAMPLE

If the C program in file f.c contains

```
char ident[] = "@(#)identification information";
```

and f.c is compiled to yield f.o and a.out, the command

```
what f.c f.o a.out
```

prints

```
f.c:
        identification information

f.o:
        identification information

a.out:
        identification information
```

DIAGNOSTICS

- Use help for explanations.

NOTES

- It's possible that an unintended occurrence of the pattern @(#) could be found by chance but it rarely causes any harm.

SEE ALSO

sccs, get, help
Concentrix-2800 Programming Tools Guide: Source Code Control System – System V Interface

whatis

show description of word

The whatis command displays a one-line synopsis of each on-line documentation file whose name matches a specified word. To read the actual on-line documentation, use the man command.

---

whatis *word...*

ARGUMENTS

word is considered separately from each other *word*. The case of letters is ignored.

NOTES

- Whatis is really just a symbolic link to man that invokes the `-f` option.

FILES

/usr/man/whatis on-line documentation database file

SEE ALSO

apropos, man, whatisin, whereis

whatisin

show contents of on-line documentation section

The whatisin command displays a one-line synopsis of each file in a specified on-line documentation section. To read the actual on-line documentation, use the man command.

---

whatisin [-] [*section* | all]

ARGUMENTS

- simulates the cat -s command for output. This causes the output to be continuous, and for multiple blank lines to be compressed to one.

By default, the output simulates the more -s command, with the display pausing after each page and multiple blank lines compressed to one.

section specifies a section of the documentation (as explained below). If no *section* is specified, whatisin displays a list of sections.

all lists a one-line synopsis of each file in all the sections of documentation.

SECTION NAMES

The mapping from "sections" to the Alliant FX/Series documentation set is:

Section	UNIX Manual	Alliant
0	n/a	overview of features and documentation
1	Commands	Concentrix-2800 Commands and Applications Manual
1b	Commands	Bourne shell
1c	Commands	C shell
1o	n/a	old commands
1u	n/a	unsupported commands
2	System calls	Concentrix-2800 System Reference Manual: Calls
3	Subroutines	Concentrix-2800 System Reference Manual: Calls
4	Special files	Concentrix-2800 System Reference Manual: Misc
5	File formats	Concentrix-2800 System Reference Manual: Misc
6	Games	not documented
7	Miscellaneous	Concentrix-2800 Document Preparation Guide, etc.
8	Maintenance	Concentrix-2800 Administration Commands
a	n/a	Alliant Libraries
ac	n/a	FX/ and FX/2800 Series Common Library
am	n/a	FX/ and FX/2800 Series Signal Processing Library
am	n/a	FX/ and FX/2800 Series Linear Algebra Library
am	n/a	FX/ and FX/2800 Series Cray Library
c	n/a	FX/C and FX/C-2800 Programmer's Handbook
f	n/a	FX/Fortran Language Manual
ff	n/a	FX/Fortran, FX/Fortran-2800 Language Manual: Format
fi	n/a	FX/Fortran, FX/Fortran-2800 Language Manual: Intrinsics
fl	n/a	FX/Fortran, FX/Fortran-2800 Language Manual: Lib Rtns
fs	n/a	FX/Fortran, FX/Fortran-2800 Language Manual: Stmnts
r	n/a	FX/ and FX/2800 Series PHIGS/PHIGS+ C & Fortran Refs

In addition, whatisin recognizes the following *section* names:

l	local	files specific to local system
n	new	files added since current software release
o	old	files from previous software release
p	public	public domain software

FILES

/usr/man/whatis	on-line documentation database file
/usr/man/whatisin	list of sections

SEE ALSO

apropos, man, whatis, whereis

whereis

locate source, binary, and on-line documentation files

The whereis command locates source, binary, and on-line documentation (man) files for specified names.

---

whereis [-sbm] [-u] [-SBM *dir*... -f] [*name*...]

whereis -d

ARGUMENTS

-sbm restricts the search to source (s), binary (b), or man (m) files.

-u restricts the search to "unusual" *names*, i.e. those that do not have one entry of each requested type.

-SBM *dir* specifies the source (S), binary (B), or man (M) directories to search. By default, whereis searches in a list of standard directories (see -d). The supplied pathnames must be absolute; i.e. each must begin with a slash.

-f terminates the last -SBM *dir*... option and signals the start of the *names*.

name is first stripped of leading pathname components and any (single) trailing extension of the form .ext, for example: .c. Prefixes of s. resulting from the use of source code control are also removed.

-d displays the standard directories searched by whereis, which are:

source

```
/usr/src/bin
/usr/src/usr.bin
/usr/src/etc
/usr/src/usr.etc
/usr/src/ucb
/usr/src/games
/usr/src/usr.lib
/usr/src/lib
/usr/src/local
/usr/src/new
/usr/src/old
/usr/src/include
/usr/src/lib/libc/gen
/usr/src/lib/libc/stdio
/usr/src/lib/libc/sys
/usr/src/lib/libc/net/common
/usr/src/lib/libc/net/inet
/usr/src/lib/libc/net/misc
/usr/src/lib/libc/compat-4.1
/usr/src/lib/libc/compat-sys5
/usr/src/lib/libc/inet
/usr/src/lib/libc/ns
```

binary

```
/etc
/bin
/usr/bin
/usr/games
/lib
/usr/ucb
/usr/lib
/usr/local
/usr/new
/usr/old
/usr/hosts
/usr/include
/usr/alliant
/usr/sys5
```

documentation

```
/usr/man/man[1-8,a,f,l,n,o,p,r]
/usr/man/cat[1-8,a,f,l,n,o,p,r]
```

source **binary** **documentation**

```
/usr/src/lib/libc/rpcsvc
/usr/src/lib/libc/yp
/usr/src/lib/libc/alliant/alliant
/usr/src/lib/libc/alliant/gen
/usr/src/lib/libc/alliant/net
/usr/src/lib/libc/alliant/stdio
/usr/src/lib/libc/alliant/sys-alliant
/usr/src/lib/libc/alliant/sys-bsd
/usr/src/lib/libc/alliant/sys-realtime
/usr/src/lib/libc/alliant/sys-system5
/usr/src/ucb/pascal
/usr/src/ucb/pascal/utilities
/usr/src/undoc
/usr/src/alliant
/usr/src/sys5
```

EXAMPLE

Find all files in the current working directory that have no on-line documentation.

```
whereis -m -u *
```

Find all files in /usr/ucb that do not have source in /usr/src/ucb:

```
cd /usr/ucb; whereis -s -u -S /usr/src/ucb -f *
```

SEE ALSO

apropos, man, whatis, whatisin
Concentrix-2800 System Administration: catman

which

show pathname for command (C shell)

The which command looks for the files that would be executed had the specified *names* been given as commands. Each *name* is expanded if it is aliased, and searched for along your path. Both aliases and path are taken from your .cshrc file unless the -c option is given.

---

which [-c] [*name...*]

ARGUMENTS

-c specifies the use of current aliases and path.

name specifies the name of a command.

FILES

~/.cshrc source of aliases and path values

DIAGNOSTICS

- A diagnostic is given for names that are aliased to more than a single word, or if an executable file with the argument name was not found in the path.

NOTES

- which must be executed by a C shell.

while

test–and–loop control structure (C shell)

The C shell while command provides a simple test-and-loop structure. It is intended for use in C shell scripts.

---

while (*expression*)
command
.
.
.
end

ARGUMENTS

expression is an numeric, filename, or command expression as described in the *Concentrix-2800 System User's Guide, C Shell Reference.*

command is any valid command.

DESCRIPTION

The loop iterates as long as the *expression* returns true (non-zero). If the *expression* is false upon reaching the while command, the loop does not iterate at all.

You can branch to the end of a while loop. The continue command causes the remaining commands on the line to execute, then transfers control to the end command, beginning another test and iteration. For example, consider the script `temp3`:

```
#
@ i = 0
while ($i <= 5)
        @ i += 1
        echo $i
        if ($i <= 3) continue
        echo test failed
end

% temp3
1
2
3
4
test failed
5
test failed
6
test failed
```

You also can branch out of a while loop. The break command causes the remaining commands on the line to execute, then transfers control to the command after the end command, terminating the loop. For example, consider the script `temp4`:

```
#
@ i = 0
while ($i <= 5)
        @ i += 1
        echo $i
        if ($i > 3) break
end

% temp4
1
2
3
4
```

NOTES

- The while and end must appear alone on their input lines.
- If the input is a terminal, prompting occurs the first time through the loop (as for the foreach command).
- The Bourne shell while command is similar in function to while.

SEE ALSO

if, if-then-else, foreach, switch

while

test–and–loop control structure (Bourne shell)

The Bourne shell while command provides a test-and-loop structure. It is intended for use in Bourne shell scripts.

---

while *expression*
do
statement
statement
done

ARGUMENTS

expression a numeric expression or command.

statement any valid command.

DESCRIPTION

The loop iterates as long as the *expression* returns true (non-zero).

If the *expression* is initially false upon reaching the while command, the loop does not iterate at all.

EXAMPLE

The Bourne shell script in this example displays the word "Hello" five times and stops.

```
counter=1
while `test $counter -le 5`
  do
  echo Hello
  counter=`expr $counter + 1`
  done
```

who

show who is on the system

The who command lists the login name, terminal name, and login time for each current system user.

---

who [*file*] [am I]

ARGUMENTS

file specifies a file to be examined (typically /usr/adm/wtmp, which contains a record of all the logins since it was created).

am I (also **'are you'**) tells who you are logged in as.

DESCRIPTION

Without an argument, who examines the /etc/utmp file to obtain its information. With an argument, *who* lists logins, logouts, and crashes since the creation of the wtmp file. Each login is listed with user name, terminal name (with /dev/ suppressed), and date and time. Logouts produce a similar line without a user name. Reboots produce a line with x in the place of the device name, and a fossil time indicative of when the system went down.

NOTES

- `Who am i` does not work if you are a superuser.

FILES

/etc/utmp

SEE ALSO

Concentrix-2800 System Reference Manual: System Calls and C Library Routines: getuid
Concentrix-2800 System Reference Manual: Miscellaneous Routines, Special Files, and File Formats: utmp

whoami

display effective current user id

The whoami command displays who you are. It works even if you are a superuser.

---

whoami

FILES

/etc/passwd	Name data base

SEE ALSO

who

whois

DARPA Internet user name directory service

Whois finds the full name of an Internet user, given a part of the name or a handle.

---

whois *name*

whois help

ARGUMENTS

name A part of the name or handle of the person you want to find.

help Tells whois to display a reminder of how to use it.

DESCRIPTION

Enter a name or a handle ("ident"), such as "Smith" or "SRI-NIC". Starting with a period forces a name-only search; starting with an exclamation point forces a handle-only search. Following are some examples:

```
Smith           ;looks for name or handle SMITH

!SRI-NIC        ;looks for handle SRI-NIC only

.Smith, John    ;looks for name JOHN SMITH only
```

Adding "..." to the argument will match anything from that point, e.g. "ZU..." will match ZUL, ZUM, etc.

To find the entire membership list of a group or organization, use an asterisk character "*" directly preceding the given argument. (CAUTION: If there are a lot of members this will take a long time!) You may of course use an exclamation point and an asterisk, or a period and an asterisk together.

SEE ALSO

RFC 812: Nicname/Whois

window

a windowing environment

Window implements a window environment on ASCII terminals.

---

window [-t] [-f] [-d] [-e *escape-char*] [-c *command*]

ARGUMENTS

-t	Turn on terse mode (see below).
-f	Fast. Don't perform any startup action.
-d	Ignore .windowrc and create the two default windows instead.
-e *escape-char*	Set the escape character to *escape-char*. *Escape-char* can be a single character; or ^X (where X is any character), meaning that the escape character will be CTRL-X.
-c *command*	Execute the string command as a long command (see below) before doing anything else.

OVERVIEW

Window implements a window environment on ASCII terminals.

A window is a rectangular portion of the physical terminal screen associated with a set of processes. Its size and position can be changed by the user at any time. Processes communicate with their window in the same way they normally interact with a terminal — through their standard input, output, and diagnostic file descriptors. The window program handles the details of redirecting input and output to and from the windows. At any one time, only one window can receive input from the keyboard, but all windows can simultaneously send output to the display.

Windows can overlap and are framed as necessary. Each window is named by one of the digits "1" to "9". This one character identifier, as well as a user definable label string, are displayed with the window on the top edge of its frame. A window can be designated to be in the foreground, in which case it will always be on top of all normal, non-foreground windows, and can be covered only by other foreground windows. A window need not be completely within the edges of the terminal screen. Thus a large window (possibly larger than the screen) may be positioned to show only a portion of its full size.

Each window has a cursor and a set of control functions. Most intelligent terminal operations such as line and character deletion and insertion are supported. Display modes such as underlining and reverse video are available if they are supported by the terminal. In addition, similar to terminals with multiple pages of memory, each window has a text buffer which can have more lines than the window itself.

PROCESS ENVIRONMENT

When window starts up, the commands (see Long Commands below) contained in the file .windowrc in the user's home directory are executed. If this file does not exist, two equal sized windows spanning the terminal screen are created by default.

With each newly created window, a shell program is spawned with its process environment tailored to that window. Its standard input, output, and diagnostic file descriptors are bound to one end of either a pseudo-terminal (pty (4)) or a UNIX domain socket (socketpair (4)). If a pseudo-terminal is used, then its special characters and modes (see stty) are copied from the physical terminal. A termcap (5) entry tailored to this window is created and passed as environment (environ (5)) variable TERMCAP. The termcap entry contains the window's size and characteristics as well as information from the physical terminal, such as the existence of underline, reverse video, and other display modes, and the codes produced by the terminal's function keys, if any. In addition, the window size attributes of the pseudo-terminal are set to reflect the size of this window, and updated whenever it is changed by the user. In particular, the editor vi uses this information to redraw its display.

OPERATION

During normal execution, window can be in one of two states: conversation mode and command mode. In conversation mode, the terminal's real cursor is placed at the cursor position of a particular window--called the current window--and input from the keyboard is sent to the process in that window. The current window is always on top of all other windows, except those in foreground. In addition, it is set apart by highlighting its identifier and label in reverse video.

Typing window's escape character (normally CTRL P) in conversation mode switches it into command mode. In command mode, the top line of the terminal screen becomes the command prompt window, and window interprets input from the keyboard as commands to manipulate windows.

There are two types of commands: short commands are usually one or two key strokes; long commands are strings either typed by the user in the command window (see the ":" command below), or read from a file (see source below).

SHORT COMMANDS

Below, # represents one of the digits "1" to "9" corresponding to the windows 1 to 9. ESC is the escape key, or ^[.

Select window # as the current window and return to conversation mode.

%# Select window # but stay in command mode.

CTRL ^ Select the previous window and return to conversation mode. This is useful for toggling between two windows.

ESC Return to conversation mode.

CTRL P Return to conversation mode and write ^P to the current window. Thus, typing two ^P's in conversation mode sends one to the current window. If

the window escape is changed to some other character, that character takes the place of ^P here.

? List a short summary of commands.

CTRL L Redraw the screen.

q Exit window. Confirmation is requested.

CTRL Z Suspend window.

w Create a new window. The user is prompted for the positions of the upper left and lower right corners of the window. The cursor is placed on the screen and the keys "h", "j", "k", and "l" move the cursor left, down, up, and right, respectively. The keys "H", "J", "K", and "L" move the cursor to the respective limits of the screen. Typing a number before the movement keys repeats the movement that number of times. Return enters the cursor position as the upper left corner of the window. The lower right corner is entered in the same manner. During this process, the placement of the new window is indicated by a rectangular box drawn on the screen, corresponding to where the new window will be framed. Typing escape at any point cancels this command.

This window becomes the current window, and is given the first available ID. The default buffer size is used (see nline command below). Only fully visible windows can be created this way.

c# Close window #. The process in the window is sent the hangup signal (see kill (1)). Csh (1) should handle this signal correctly and cause no problems.

m# Move window # to another location. A box in the shape of the window is drawn on the screen to indicate the new position of the window, and the same keys as those for the w command are used to position the box. The window can be moved partially off-screen.

M# Move window # to its previous position.

s# Change the size of window #. The user is prompted to enter the new lower right corner of the window. A box is drawn to indicate the new window size. The same keys used in w and m are used to enter the position.

S# Change window # to its previous size.

CTRL Y Scroll the current window up by one line.

CTRL E Scroll the current window down by one line.

CTRL U Scroll the current window up by half the window size.

CTRL D Scroll the current window down by half the window size.

CTRL B Scroll the current window up by the full window size.

CTRL F Scroll the current window down by the full window size.

h	Move the cursor of the current window left by one column.
j	Move the cursor of the current window down by one line.
k	Move the cursor of the current window up by one line.
l	Move the cursor of the current window right by one column.
CTRL S	Stop output in the current window.
CTRL Q	Start output in the current window.
:	Enter a line to be executed as long commands. Normal line editing characters (erase character, erase word, erase line) are supported.

LONG COMMANDS

Long commands are a sequence of statements parsed much like a programming language, with a syntax similar to that of C. A statement is either an expression or a conditional.

The character "#" begins a comment which terminates at the end of the line.

DATA TYPES

There are two data types: string and number.

A string is a sequence of letters or digits beginning with a letter. "_" (underscore) and "." (period) are considered letters. Alternately, non-alphanumeric characters can be included in strings by quoting them within " " " (double quotes) or escaping them with "\" (backslash). In addition, the "\" sequences of C are supported, both inside and outside quotes (e.g., "\n" is a new line, "\r" a carriage return). For example, these are legal strings: abcde01234, "&#$^*&#", ab"$#"cd, ab\$\#cd, "/usr/ucb/window".

A number is an integer value in one of three forms: a decimal number, an octal number preceded by "0", or a hexadecimal number preceded by "0x" or "0X". The natural machine integer size is used (i.e., the signed integer type of the C compiler). As in C, a non-zero number represents a Boolean true.

EXPRESSIONS

Expressions in window are similar to those in the C language, with most C operators supported on numeric operands. In addition, some are overloaded to operate on strings.

Expressions are terminated with a new line or ";" (semicolon). To continue an expression on the next line, terminate the first line with "\" (backslash). String and numeric expressions and variables are supported.

When an expression is used as a statement, its value is discarded after evaluation. Therefore, only expressions with side effects (assignments and function calls) are useful as statements.

Single valued (no arrays) variables are supported, of both numeric and string values. Some variables are predefined (see below).

The operators in order of increasing precedence are:

expr1 = expr2

Assignment. The variable of name *expr1*, which must be string valued, is assigned the result of *expr2*. Returns the value of *expr2*.

expr1 ? expr2 : expr3

Returns the value of *expr2* if *expr1* evaluates true (non–zero numeric value); returns the value of expr3 otherwise. Only one of *expr2* and expr3 is evaluated. *Expr1* must be numeric.

expr1 || expr2

Logical or. Numeric values only. Short circuit evaluation is supported (i.e., if *expr1* evaluates true, then *expr2* is not evaluated).

expr1 && expr2

Logical and with short circuit evaluation. Numeric values only.

expr1 | expr2

Bitwise or. Numeric values only.

expr1 ^ expr2

Bitwise exclusive or. Numeric values only.

expr1 & expr2

Bitwise and. Numeric values only.

expr1 == expr2, expr1 != expr2

Comparison (equal and not equal, respectively). The boolean result (either 1 or 0) of the comparison is returned. The operands can be numeric or string valued. One string operand forces the other to be converted to a string in necessary.

expr1 < expr2, expr1 > expr2

Less than, greater than, less than or equal to, greater than or equal to. Both numeric and string values, with automatic conversion as above.

expr1 << expr2, expr1 >> expr2

If both operands are numbers, *expr1* is bit shifted left (or right) by *expr2* bits. If *expr1* is a string, then its first (or last) *expr2* characters are returns (if *expr2* is also a string, then its length is used in place of its value).

expr1 + *expr2*, *expr1* – *expr2*

Addition and subtraction on numbers. For "+", if one argument is a string, then the other is converted to a string, and the result is the concatenation of the two strings.

expr1 * *expr2*, *expr1* / *expr2*

Multiplication, division, modulo. Numbers only.

–*expr*, -*expr*, !*expr*, $*expr*, $?*expr*

The first three are unary minus, bitwise complement and logical complement on numbers only. The operator, "$", takes *expr* and returns the value of the variable of that name. If *expr* is numeric with value n and it appears within an alias macro (see below), then it refers to the nth argument of the alias invocation. "$?" tests for the existence of the variable *expr*, and returns 1 if it exists or 0 otherwise.

expr(*arglist*)

Function call. *Expr* must be a string that is the unique prefix of the name of a built–in window function or the full name of a user defined alias macro. In the case of a built–in function, *arglist* can be in one of two forms: *expr1*, *expr2*, . . . *argname1* = *expr1*, *argname2* = *expr2*, . . .

The two forms can in fact be intermixed, but the result is unpredictable. Most arguments can be omitted; default values will be supplied for them. The argnames can be unique prefixes of the the argument names. The commas separating arguments are used only to disambiguate, and can usually be omitted.

Only the first argument form is valid for user defined aliases. Aliases are defined using the alias built–in function (see below). Arguments are accessed via a variant of the variable mechanism (see "$" operator above).

Most functions return value, but some are used for side effect only and so must be used as statements. When a function or an alias is used as a statement, the parenthesis surrounding the argument list may be omitted. Aliases return no value.

BUILT–IN FUNCTIONS

The arguments are listed by name in their natural order. Optional arguments are in square brackets ("[ ]").

alias([*string*], [*string–list*])

If no argument is given, all currently defined alias macros are listed. Otherwise, *string* is defined as an alias, with expansion *string–list*. The previous definition of *string*, if any, is returned. Default for *string–list* is no change.

close(*window–list*)

Close the windows specified in *window–list*. If *window–list* is the word **all**, than all windows are closed. No value is returned.

cursormodes([*modes*])

Set the window cursor to *modes*. *Modes* is the bitwise or of the mode bits defined as the variables mul (underline), mrev (reverse video), mblk (blinking), and mgrp (graphics, terminal dependent). Return value is the previous modes. Default is no change. For example, cursor($mrev|$mblk) sets the window cursors to blinking reverse video.

echo([*window*], [*string-list*])

Write the list of strings, *string-list*, to *window*, separated by spaces and terminated with a new line. The strings are only displayed in the window, the processes in the window are not involved (see write below). No value is returned. Default is the current window.

escape([*escapec*])

Set the escape character to *escapec*. Returns the old escape character as a one character string. Default is no change. *Escapec* can be a string of a single character, or in the form ^X, meaning CTRL X.

foreground([*window*], [*flag*])

Move *window* in or out of foreground. *Flag* can be one of **on**, **off**, **yes**, **no**, **true**, or **false**, with obvious meanings, or it can be a numeric expression, in which case a non-zero value is true. Returns the old foreground flag as a number. Default for *window* is the current window, default for *flag* is no change.

label([*window*], [*label*])

Set the label of *window* to *label*. Returns the old label as a string. Default for *window* is the current window, default for *label* is no change. To turn off a label, set it to an empty string ("").

list()

No arguments. List the identifiers and labels of all windows. No value is returned.

nline([*nline*])

Set the default buffer size to *nline*. Initially, it is 48 lines. Returns the old default buffer size. Default is no change. Using a very large buffer can slow the program down considerably.

select([*window*])

Make *window* the current window. The previous current window is returned. Default is no change.

shell([*string-list*])

Set the default window shell program to *string-list*. Returns the first string in the old shell setting. Default is no change. Initially, the default shell is taken from the environment variable SHELL.

source(*filename*)

Read and execute the long commands in *filename*. Returns −1 if the file cannot be read, 0 otherwise.

terse([*flag*])

Set terse mode to *flag*. In terse mode, the command window stays hidden when in command mode, and errors are reported by sounding the terminal's bell. *Flag* can take on the same values as in foreground above. Returns the old terse flag. Default is no change.

unalias(*alias*)

Undefine *alias*. Returns −1 if *alias* does not exist, 0 otherwise.

unset(*variable*)

Undefine *variable*. Returns −1 if variable does not exist, 0 otherwise.

variables()

No arguments. List all variables. No value is returned.

window([*row*], [*column*], [*nrow*], [*ncol*], [*nline*], [*frame*], [*pty*], [*mapnl*], [*shell*])

Open a window with upper left corner at *row*, *column* and size *nrow*, *ncol*. If *nline* is specified, then that many lines are allocated for the text buffer. Otherwise, the default buffer size is used. Default values for *row*, *column*, *nrow*, and *ncol* are, respectively, the upper, left-most, lower, or right-most extremes of the screen.

Frame, *pty*, and *mapnl* are flag values interpreted in the same way as the argument to foreground (see above); they mean, respectively, put a frame around this window (default true), allocate pseudo terminal for this window rather than socketpair (default true), and map new line characters in this window to carriage return and line feed (default true if socketpair is used, false otherwise). *Shell* is a list of strings that will be used as the shell program to place in the window (default is the program specified by shell, see below). The created window's identifier is returned as a number.

write([*window*], [*string-list*])

Send the list of strings, *string-list*, to *window*, separated by spaces but not terminated with a new line. The strings are actually given to the window as input. No value is returned. Default is the current window.

CONDITIONAL STATEMENT

Window has a single control structure, the fully bracketed if–statement, in the form:

```
if <expr> then
     <statement>
     . . .
elsif <expr> then
     <statement>
     . . .
else
     <statement>
     . . .
endif
```

The else and elsif parts are optional, and the latter can be repeated any number of times. <Expr> must be numeric.

PREDEFINED VARIABLES

These variables are for information only. Redefining them does not affect the internal operation of window.

baud The baud rate as a number between 50 and 38400.

modes The display modes (reverse video, underline, blinking, graphics) supported by the physical terminal. The value of modes is the bitwise or of some of the one bit values, mblk, mgrp, mrev, and mul (see below). These values are useful in setting the window cursors' modes (see cursormodes above).

mblk The blinking mode bit.

mgrp The graphics mode bit (not very useful).

mrev The reverse video mode bit.

mul The underline mode bit.

ncol The number of columns on the physical screen.

nrow The number of rows on the physical screen.

term The terminal type. The standard name, found in the second name field of the terminal's TERMCAP entry, is used.

FILES

~/.windowrc	startup command file
/dev/[pt]ty[pq]?	pseudo-terminal devices.

write

write to another user

The write command copies lines from your terminal to that of another user.

---

write *user* [*ttyname*]

DESCRIPTION

user specifies a user name.

ttyname specifes a terminal name (useful for writing to a user who is logged in more than once).

ARGUMENTS

When first called, write sends the message

```
Message from yourname@yoursystem yourttyname...
```

The recipient of the message should write back at this point. Communication continues until an end of file is read from the terminal or an interrupt is sent. At that point write writes EOT on the other terminal and exits.

Permission to write can be denied or granted by use of the mesg command. At the outset writing is allowed. Certain commands, in particular nroff and pr disallow messages in order to prevent messy output.

If an exclamation point (!) is found at the beginning of a line, write calls the shell to execute the rest of the line as a command.

The following protocol is suggested for using write: when you first write to another user, wait for him to write back before starting to send. Each party should end each message with a distinctive signal. (o) for 'over' means that the other can reply. (oo) for "over and out" is suggested when conversation is about to be terminated.

FILES

/etc/utmp	to find user
/bin/sh	to execute '!'

SEE ALSO

mesg, who, mail, talk

xget

read encrypted mail

The xget command reads encrypted mail. It asks for your password, then gives you the messages.

---

xget

DESCRIPTION

Xget, along with xsend and enroll, implement a secure communication channel similar to mail except that no one can read the messages except the intended recipient. The method embodies a public-key cryptosystem using knapsacks.

FILES

/usr/spool/secretmail/*.key	keys
/usr/spool/secretmail/*.[0–9]	messages

SEE ALSO

xsend, enroll, mail

xproc

process time and cpu monitor

The xproc command gives a continuous display of processes (similar to ps) or process times (user/system, IP, CE and COMPLEX), or activity on the individual processors. Unless the -i option is used, the display is updated every second.

---

xproc [-cpt] [-i*n*] [-1*n*]

ARGUMENTS

-c displays processor activity.

-p displays processes. This is the default.

-t displays process times.

-i*n* (optional) causes the display to pause *n* seconds between display updates. If the option is not specified, the display is updated every second. A value of 0 updates the display continuously.

-1*n* (optional) restricts the number of lines of displayed data to *n*. If the option is not specified, the display includes at least 8 lines.

DESCRIPTION

You can change the display while it is running by typing any of the above option charcacters with the exception of 1*n*. You can change the interval by typing the appropriate digit from 0 to 9.

The following interactive dsiplay options are also available:

^L	Refresh the display.
?	Display a help window.
u	Move marker up one line.
U	Move marker up ten lines.
d	Move marker down one line.
D	Move marker down ten lines.
b	Move back one screen.
f	Move forward one screen.
s	Go to first line.
e	Go to last line.
m	Mark a line to follow. (Not available for CPU activity.)
q,Q	Quit out of xproc.

The various headings have the following meanings:

USER — The user name of the process owner.
PID — The process ID number.
STATE — The state the process is in ... one of SLEEP, WAIT, RUN, IDLE, ZOMBY, or STOP.
RSZ — The resident set size of the process.
WSZ — The working set size of the process.
PRI — The process priority.
NI — The process 'nice' level.
CL — The current class the process is running at.
PR — The run queue index that the process is at.
CPU — Which processor the process is/can run on.
FLAGS — Memory management flags for the process ... one or more of SYS, NOMEM, SWAPPED, LOCK, NOSWAP and NOPAGE.
IP,CE,CMPLX — Which resource the accumulated time is displayed for. This is followed by the number of processors the process is running on (in ()), if applicable.
CLASS LIST — The list of classes (and their priority) the CPU will run. These are the classes defined by setsched.

When displaying CPU activity, the state of the relevant complex, along with the concurent and detach thresholds, are shown immediately below the activity line.

EXAMPLE

The following example shows the default display.

```
% xproc -i9

PROC slots .. Used = 174   Free = 358              Elapsed Time = 0.00s
Files ..... Open = 440   Free = 1550
COMMAND            PID USER     STATE    RSZ    WSZ PRI NI CL PR CPU  FLAGS
cc -c -Og -I../. 27795 ama      STOP     56K   128K  30 0  3  0 IpCe
fxc1 -Og -I../.. 27855 ama      STOP    212K   4.5M  30 0  3  1 IpCe
sh -ce for i in  27399 eaw      SLEEP    32K   128K  30 0  3  0 IpCe
make JFLAG=-j5   27997 eaw      SLEEP   184K   308K  30 0  3  0 IpCe
rlogind           6255 root     SLEEP   160K   256K  26 0  3  0 IpCe
-csh             29164 hossein  SLEEP   200K   432K  40 0  3  0 IpCe
tail -f meme     29522 michaud  SLEEP    32K    68K  40 0  3  0 IpCe
rlogin flash     26995 shtil    SLEEP   112K   116K  28 0  3  0 IpCe
rlogin flash     31238 shtil    SLEEP   112K   116K  26 0  3  0 IpCe
vi Motif.tmpl    31615 ama      SLEEP   140K   448K  28 0  3  0 IpCe
vi site.def      31888 ama      SLEEP   144K   448K  28 0  3  0 IpCe
rlogind          30847 root     SLEEP   160K   256K  26 0  3  0 IpCe
-csh             31450 perron   SLEEP   124K   380K  40 0  3  0 IpCe
csh ./maketape.s 32181 hossein  SLEEP   192K   428K  40 0  3  0 IpCe
tar cvf /dev/rsc   115 hossein  SLEEP   100K   112K  20 0  3  1 IpCe
```

FILES

/dev/kmem

SEE ALSO

mon, ps, kconfig

xsend

send encrypted mail

The xsend command sends encrypted mail in the same manner as the ordinary mail command (except that it accepts only one *person*). A message announcing the receipt of encrypted (secret) mail is also sent by ordinary mail.

---

xsend *person*

ARGUMENTS

person specifies the person to receive the mail.

DESCRIPTION

Xsend, along with xget and enroll, implement a secure communication channel similar to mail except that no one can read the messages except the intended recipient. The method embodies a public-key cryptosystem using knapsacks.

NOTES

- Encrypted mail should be integrated with ordinary mail. The announcement of encrypted mail makes traffic analysis possible.

FILES

/usr/spool/secretmail/*.key	keys
/usr/spool/secretmail/*.[0-9]	messages

SEE ALSO

xget, enroll, mail

xstr

extract strings from C programs

The xstr command maintains a file strings into which strings in component parts of a large program are hashed. These strings are replaced with references to this common area. This serves to implement shared constant strings, most useful if they are also read-only.

---

xstr [-c] [-] [*file*]

ARGUMENTS

-c specifies that *file* is a C source program.

- reads from the standard input.

file varies (see description).

DESCRIPTION

The command:

```
xstr -c name
```

extracts the strings from the specified C source, replacing string references by expressions of the form (&xstr[number]) for some number. An appropriate declaration of *xstr* is prepended to the file. The resulting C source is placed in the file x.c to be compiled. The strings from this file are placed in the strings data base file if they are not there already. Repeated strings and strings that are suffices of existing strings do not cause changes to the data base.

After all components of a large program have been compiled, a file xs.c declaring the common *xstr* space can be created by a command of the form:

```
xstr
```

The file xs.c should then be compiled and loaded with the rest of the program. If possible, the array can be made read-only (shared) saving space and swap overhead.

Xstr can also be used on a single file. A command

```
xstr name
```

creates files x.c and xs.c as before, without using or affecting any strings file in the same directory.

It may be useful to run xstr after the C preprocessor if any macro definitions yield strings or if there is conditional code that contains strings that may not, in fact, be needed. For example:

```
cc -E name.c | xstr -c -
cc -c x.c
mv x.o name.o
```

Xstr does not touch the file `strings` unless new items are added, thus *make* can avoid remaking `xs.o` unless truly necessary.

NOTES

- If a string is a suffix of another string in the data base, but the shorter string is seen first by xstr both strings are placed in the data base, when just placing the longer one there would do.

FILES

strings	data base of strings
x.c	massaged C source
xs.c	C source for definition of array 'xstr'
/tmp/xs*	temp file when 'xstr name' doesn't touch strings

SEE ALSO

mkstr

yacc

yet another compiler-compiler

The yacc command converts a context-free grammar into a set of tables for a simple automaton that executes an LR(1) parsing algorithm. The grammar may be ambiguous; specified precedence rules are used to break ambiguities.

---

yacc [-vd] *grammar*

ARGUMENTS

-v prepares the file `y.output` which contains a description of the parsing tables and a report on conflicts generated by ambiguities in the grammar.

-d generates the file y.tab.h with the *define* statements that associate the yacc-assigned "token codes" with the user-declared "token names". This allows source files other than y.tab.c to access the token codes.

DESCRIPTION

The output file, y.tab.c, must be compiled by the C compiler to produce a program yyparse. This program must be loaded with the lexical analyzer program, yylex, as well as *main* and *yyerror*, an error handling routine. These routines must be supplied by the user; Lex is useful for creating lexical analyzers usable by yacc.

DIAGNOSTICS

- The number of reduce-reduce and shift-reduce conflicts is reported on the standard output; a more detailed report is found in the y.output file. Similarly, if some rules are not reachable from the start symbol, this is also reported.

NOTES

- Because file names are fixed, at most one yacc process can be active in a given directory at a time.

FILES

y.output	
y.tab.c	
y.tab.h	defines for token names
yacc.tmp, yacc.acts	temporary files
/usr/lib/yaccpar	parser prototype for C programs

SEE ALSO

lex
Concentrix-2800 Programming Tools Guide: YACC – Yet Another Compiler Compiler
LR Parsing, A. V. Aho and S. C. Johnson, Computing Surveys, June, 1974.

yes

be repetitively affirmative

The yes command repeatedly outputs y, which can be used as the standard input to programs that require repetitive confirmation. To terminate yes, press CTRL C.

---

yes [*expletive*]

ARGUMENTS

expletive specifies a string to output repeatedly.

ypcat

display values in an NIS data base

The ypcat command displays the values in a specified NIS map.

---

ypcat [-a] [-k] [-t] [-d *domainname*] *mname*

ypcat -x

ARGUMENTS

-a gets all the key_value pairs back from the NIS, including private symbols. These are uninteresting to almost everyone.

-k displays the keys for those maps in which the values are null or the key is not part of the value.

-t inhibits translation of abbreviated mapname to mapname. For example, ypcat -t passwd fails because there is no file named passwd, while ypcat passwd is translated to ypcat passwd.byname. The purpose of this option is to let you specify a mapname that is also a keyword name.

-x Display the map nickname table. This lists the nicknames the command knows of, and indicates the mapname associated with each nickname.

-d *domainname*
specifies a domain other that the default domain, which is returned by the domainname command. The default domain is the one that is interesting to most users.

mname can be either a *mapname* or an map nickname. Since ypcat employs NIS service, the location of the NIS server cannot be supplied.

DESCRIPTION

ypcat prints out values in a Network Information Service (NIS) map specified by *mname*, which may be either a mapname or a map nickname. Since ypcat uses the NIS service, no NIS server is specified.

Refer to ypfiles and ypserv for an overview of the NIS service, and to the *Concentrix Network File System User's Guide* for more complete information.

Suppose you want to look at the system password file. Each machine has a file /etc/passwd, containing only a small amount of information. The network-wide password database is served by the NIS, and holds information in a map named passwd.byname, whose abbreviated name is passwd. Thus, to see the system passwords, type:

```
ypcat passwd
```

Here is a list of abbreviated mapnames, full mapnames, and corresponding system files:

passwd	passwd.byname	/etc/passwd
group	group.byname	/etc/group
hosts	hosts.byaddr	/etc/hosts
networks	networks.byaddr	/etc/networks
services	services.byname	/etc/services
protocols	protocols.bynumber	/etc/protocols
netgroup	netgroup	/etc/netgroup

A list of all maps in the default domain can be seen by typing:

```
ypcat -k ypmaps
```

A list of all the known domains can be seen by typing:

```
ypcat -d yp_private -k ypdomains
```

SEE ALSO

domainname, ypmatch
Concentrix-2800 System Administration: ypserv, yppush
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: ypfiles
Concentrix Network File System User's Guide, chapters on NIS administration, protocols

ypmatch

print the value of one or more keys from a yp map

The ypmatch command prints the values associated with one or more keys from the NIS (YP) map specified by a *mname*, which may be either a *mapname* or a map *nickname*.

---

ypmatch [-d *domain* **] [-k] [-t]** *key ... mname*
ypmatch -x

ARGUMENTS

-d Specify a domain other that the default domain.

-k Before printing the value of a key, print the key itself, followed by a colon (':'). This is useful only if the keys are not duplicated in the values, or you've specified so many keys that the output could be confusing.

-t Inhibit translation of nickname to mapname. For example, **ypmatch -t zippy passwd** will fail because there is no map named "passwd". **Ypmatch zippy passwd** will be translated to "ypmatch zippy passwd.byname".

-x Display the map nickname table. This lists the nicknames (mnames) the command knows of, and indicates the *mapname* associated with each nickname.

DESCRIPTION

Ypmatch prints the values associated with one or more keys from the network information service (YP) map specified by a *mname*, which may be either a *mapname* or a map *nickname*.

Multiple keys can be specified; the same map will be searched for all . The keys must be exact values insofar as capitalization and length are concerned. No pattern matching is available. If a key is not matched, a diagnostic message is produced.

SEE ALSO

ypcat,
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: ypfiles
Concentrix Network File System User's Guide, chapters on NIS administration, protocols

yppasswd

change login password in network information service

The yppasswd command changes (or installs) a password in the NIS database, which can be different from the password on your own machine.

---

yppasswd [*name*]
ypchfn [*name*]
ypchsh [*name*]

ARGUMENTS

name specifies a user (default is your own name).

DESCRIPTION

Yppasswd changes (or installs) a network password associated with the user *name* (your own name by default) in the NIS database. The NIS password may be different from the one on your own machine.

Ypchfn changes the full name associated with *name* in the NIS database.

Ypchsh changes the login shell associated with *name* in the NIS database.

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

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

Only the owner of the name or the super-user can change a password; in either case you must prove you know the old password.

The NIS password daemon, yppasswdd must be running on your NIS server in order for the new password to take effect.

NOTES

- The update protocol passes all the information to the server in one rpc call, without ever looking at it. Thus if you type in your old password incorrectly, you are not notified until after you have entered your new password.

SEE ALSO

passwd
Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: ypfiles
Concentrix-2800 System Administration: yppasswdd
Concentrix Network File System User's Guide, chapters on NIS administration, protocols

ypwhich

which host is the YP server or map master?

The ypwhich command tells which YP server supplies NIS services to a YP client, or which is the master for a map.

---

ypwhich [**-d** [*domain*]] [**-V1** | **-V2**] [*hostname*]
ypwhich [**-t** *mapname*] [**-d** *domain*] **-m** [*mname*]
ypwhich -x

ARGUMENTS

-d	Use *domain* instead of the default domain.
-V1	Which server is serving v.1 YP protocol–speaking client processes?
-V2	Which server is serving v.2 YP protocol client processes?
	If neither version is specified, ypwhich attempts attempts to locate the server that supplies the (current) v.2 services. If there is no v.2 server currently bound, ypwhich then attempts to locate the server supplying the v.1 services. Since YP servers and YP clients are both backward compatible, the user seldom need be concerned about which version is currently in use.
-t *mapname*	Inhibit nickname translation; useful if there is a mapname identical to a nickname. This is not true of any Sun-supplied map.
-m	Find the master YP server for a map. No *hostname* can be specified with -m. *mname* can be a mapname, or a nickname for a map. When *mname* is omitted, produce a list available maps.
-x	Display the map nickname table. This lists the nicknames (mnames) the command knows of, and indicates the *mapname* associated with each nickname.

DESCRIPTION

Ypwhich tells which YP server supplies NIS services to a YP client, or which is the master for a map. If invoked without arguments, it gives the YP server for the local machine. If *hostname* is specified, that machine is queried to find out which YP master it is using.

Refer to ypfiles(5) and ypserv(8) for an overview of the NIS.

SEE ALSO

Concentrix-2800 System Reference: Miscellaneous Routines, Special Files, and File Formats: ypfiles
Concentrix-2800 System Administration: rpcinfo, ypset, ypserv
Concentrix Network File System User's Guide, chapters on NIS administration, protocols

zcat

expand compressed files to standard output

The zcat command expands files (to standard output) that were compressed using the *compress*(1) utility.

---

zcat [*name...*]

ARGUMENTS

name the input file that you want to expand. The input file must have been compressed by the compress utility, and must have a .Z extension. If no input filename is given, the standard input is expanded to the standard output.

DESCRIPTION

Zcat expands files that have been compressed with the adaptive Lempel–Ziv coding of the compress utility. Zcat writes the uncompressed version of the file to standard output. The input files are not altered.

Every file that is expanded must have a .Z extension. If a filename that you supply does not end in .Z, zcat appends a .Z to the name you supply, before looking for the file.

Zcat is equivalent to uncompress –c. If you want to create an uncompressed version of a file (rather than just write the uncompressed data to standard output), you should use the uncompress command.

DIAGNOSTICS

- Usage: compress [–fvc] [file ...]. You specified invalid options on the command line.
- *name*: not in compressed format. The file specified to uncompress has not been compressed.
- *name*: compressed with xx bits, can only handle yy bits. File was compressed by a program that could deal with more bits than the compress code on this machine. Recompress the file with smaller bits.
- uncompress: corrupt input. A SIGSEGV violation was detected which usually means that the input file is corrupted.

NOTES

- Zcat should be more flexible about the existence (or nonexistence) of the '.Z' suffix on compressed files.

FILES

*.Z	compressed file created by compress, read by zcat.

SEE ALSO

compress, uncompress

Variations on a Theme of Huffman, Gallager, Robert G.,
I.E.E.E. Transactions on Information Theory, vol. IT-24, no. 6,
November 1978, pp. 668 – 674.

Arguments and Subcommands

(Bourne shell), positional parameters, 423

&&, I/O redirection, 85

%
  See also percent_sign
  kill, C shell, 234

+
  chmod, 63
  edit, 128
  ex, 149
  leave, 240
  lpq, 256
  more, 298
  nice, 310
  page, 323
  sort, 427
  tail, 453
  uniq, 517

+/
  more, 298
  page, 323

+linenum, GNU emacs, 133

-
  calendar, 44
  chmod, 63
  colcrt, 69
  csh, 83
  ed, 127
  edit, 128
  ex, 149
  kill
    Bourne shell, 233
    C shell, 234
  lprm, 261
  man, 285
  mkstr, 291
  more, 298
  mv, 305
  nice, 310
  od, 318
  onintr, 322
  page, 323
  rm, 377
  sh, 414
  sort, 427
  strings, 438
  tail, 453
  tip, 489
  uux, 533
  val, 538
  vi, 540
  xstr, 576

-#, lpr, 258

0...9, tar, 457

-1, ls, 266

-2, colcrt, 69

-8, rlogin, 375

-A
  gprof, 199
  ld, 237
  ls, 267
  netstat, 306

-a
  addbib, 2
  admin, 6
  apply, 13
  checknr, 58
  ctags, 94
  du, 125
  get, 191
  gprof, 199
  graph, 202
  join, 229
  lint, 244
  lprof, 263
  ls, 266
  mon, 293
  netstat, 306
  prs, 338
  qstat, 350
  qsub, 351
  refer, 367
  ruptime, 389
  rwho, 392
  script, 402
  spline, 435
  style, 444
  tee, 475
  test, 482
  troff, 496
  unexpand, 513
  ypcat, 580

a
  ar, 16
  chmod, 63
  ps, 342

-ac, execute, 152

-atime, find, 167

-B
  ctags, 94
  refer, 368
  whereis, 553

B, tar, 457

-b
  cat, 48
  col, 68
  diff, 116
  egrep, 130
  fgrep, 165
  get, 191
  gprof, 199
  graph, 202
  grep, 204
  ld, 237
  lint, 244
  refer, 367
  sort, 427
  spell, 431
  troff, 496
  whereis, 553

b
  ar, 16
  tail, 453
  tar, 457

c, ar, 16

-batch, GNU emacs, 133

-C
  lpr, 258
  ls, 267
  make, 277
  tar, 456

-c
  bc, 30
  checknr, 58
  comb, 71
  csh, 83
  diff, 115
  egrep, 130
  execute, 152
  fgrep, 165

get, 190
gprof, 199
grep, 204
install, 227
lint, 244
lpr, 259
ls, 266
mon, 293
more, 298
msgs, 301
page, 323
refer, 367
sh, 414
sort, 427
touch, 493
tr, 494
uniq, 517
xstr, 576

c
tail, 453
tar, 456

-c option, scc, 394

-cbreak, stty, 440

-ce, execute, 152

-cl, execute, 152

-crterase, stty, 442

-crtkill, stty, 442

-ctlecho, stty, 442

-D
as, 17
diff, 116
graph, 202
ld, 237

-d
admin, 6
eqn, 144
ftp, 181
GNU emacs, 133
ld, 237
look, 253
lpr, 258
ls, 266
mail, 272
make, 277
more, 298
neqn, 309
page, 323
prs, 338
sort, 427
spell, 431
spellout, 434
test, 481
tr, 494
uniq, 517
whereis, 553
yacc, 578
ypcat, 580

d, ar, 15

-decctlq, stty, 442

-E, gprof, 199

-e
admin, 6
cat, 48
csh, 83
diff, 115
diff3, 118
egrep, 130
fsplit, 180
get, 190
gprof, 199
grep, 204
join, 229
ld, 237
lprof, 263
make, 277
nroff, 316
prs, 338
refer, 367
rlogin, 375
sed, 403
sh, 414
style, 444

-echo, stty, 440

-eq, test, 481

-even, stty, 440

-exec, find, 167

-F
awk, 26
ctags, 94
gprof, 199
ld, 237
ls, 267
troff, 497

-f
admin, 4
awk, 26
checknr, 58
chgrp, 61
col, 68
csh, 83
diff, 116
egrep, 130
eqn, 144
fgrep, 165
GNU emacs, 133
gprof, 199
lex, 241
look, 253
lpr, 259
lprof, 263
ls, 266
mail, 272
make, 277, 282
man, 285
mon, 293
more, 298
msgs, 301
mt, 303
mv, 305
neqn, 309
nm, 312
page, 323
pr, 334
refer, 367
rm, 377
sed, 403
sort, 427
test, 481
touch, 493
troff, 496
whereis, 553

f
dired, 120
tail, 453
tar, 457

-flusho, stty, 443

-g
delta, 108
ftp, 181
get, 191
install, 227
lpr, 259
lprof, 263
ls, 266

g
chmod, 63
ps, 342

-group, find, 167

-h
admin, 6
diff, 116
graph, 202
history, 212
lint, 244
login option, superuser, 249
lpr, 259
lprof, 263
mail, 272
msgs, 301
netstat, 306
nroff, 316
pr, 334
source, 430
spell, 431
test, 481
w, 543

h, tar, 457

-I
adb, 1

as command, 17
dbx, 99
error, 145
lpr, 258
make, 277

-i
admin, 4
afio option, 8
bfio option, 33
cp, 80
csh, 83
dbx, 99
fgrep, 165
ftp, 181
GNU emacs, 133
get, 191
grep, 204
ld, 237
lpr, 258
ls, 266
mail, 272
make, 277, 282
mon, 293
mv, 305
netstat, 306
nroff, 316
rm, 377
sh, 414
sort, 427
tee, 475
troff, 496
ul, 506

i, ar, 16

-inline, scc command option, 394

-inum, find, 167

-j, join, 229

-k
Bourne shell, 414
get, 191
make, 277, 282
man, 285
qdel, 349
refer, 367
spline, 435
ypcat, 580

-kill, GNU Emacs option, 133

-L, ls, 267

-l
as, 17
bc, 30
cmp, 67
diff, 115
dirs, 123
edit, 128
egrep, 130
ex, 149
fgrep, 165
finger, 170
GNU emacs, 133
get, 191
grep, 204
jobs, 228
kill
Bourne shell, 233
C shell, 234
ld, 237
lpq, 256
lpr, 258
lprof, 263
ls, 266
make, 277
more, 298
msgs, 301
page, 323
pr, 334
prs, 338
qstat, 350
refer, 367
rlogin, 375
rsh, 385
ruptime, 389
style, 444
w, 543

l
ar, 16
ps, 342
tail, 453
tar, 457

-lcase, stty, 441

-links, find, 167

-litout, stty, 443

-load, GNU emacs, 133

-M
as command, 17
ld, 238
whereis, 553

-m
admin, 6
cdc, 54
delta, 108
finger, 170
get, 191
graph, 202
install, 227
ld, 238
lpr, 259
mon, 293
netstat, 306
nroff, 316
pr, 334
roffbib, 384
sort, 427
troff, 496
uusend, 532
val, 538
whereis, 553

m
ar, 15
tar, 457

-mdmbuf, stty, 443

-me, qsub, 351

-ml, style, 444

-mm, style, 444

-mu, qsub, 351

-N, mail, 272

-n, 312
admin, 4
cat, 48
csh, 83
delta, 108
echo, 126
egrep, 130
error, 145
fgrep, 165
ftp, 181
get, 191
grep, 204
lex, 241
lint, 244
lpr, 258
mail, 272
make, 278, 282
netstat, 306
nroff, 316
prelf command option, 335
refer, 367
rsh, 385
sed, 403
sh, 414
sort, 427
spline, 435
split, 436
tabs, 452
test, 481
troff, 496

n, mesg, 289

-nac, execute, 152

-name, find, 167

-nc, ld, 238

-ndc, execute, 153

-newer, find, 167

-nl, stty, 441

-no-init-file, GNU emacs, 134

-nohang, stty, 443

-o
  afio option, 8
  as, 17
  bfio option, 33
  comb, 71
  install, 227
  join, 229
  ld, 238
  msgs, 301
  nm, 312
  nroff, 316
  qsub, 351
  sort, 428
  strings, 438
  test, 482
  troff, 496

o
  ar, 16
  chmod, 63
  tar, 456

-odd, stty, 441

-ok, find, 167

-P
  lpq, 256
  lpr, 258
  lprm, 261
  refer, 368
  style, 444

-p
  addbib, 2
  bfio option, 33
  comb, 71
  delta, 109
  eqn, 144
  finger, 170
  get, 191
  lint, 244
  lpr, 258
  make, 278
  man, 285
  msgs, 301
  neqn, 309
  netstat, 306
  nm, 312
  qsub, 351
  refer, 367
  sccsdiff, 401
  spline, 435
  style, 444
  tc, 460

p
  ar, 15
  tar, 456

-p afio option, 8, 33

-pendin, stty, 443

-perm, find, 167

-pN, troff, 497

-print, find, 167

-Q, roffbib, 384

-q
  error, 145
  GNU emacs, 134
  ls, 266
  make, 278
  msgs, 301
  nroff, 316
  qsub, 351
  troff, 496

q, ar, 15

-R
  edit, 128
  ex, 149
  ls, 267

-r
  admin, 4
  cdc, 54
  cp, 80
  dbx, 99
  delta, 108
  diff, 115
  dired, 120
  edit, 128
  ex, 149
  get, 190
  graph, 202
  history, 212
  ld, 238
  lpr, 259
  ls, 266
  mail, 272
  make, 278, 282
  mon, 294
  netstat, 306
  nm, 312
  nroff, 316
  prs, 338
  qsub, 351
  rcp, 358
  rm, 377
  rmdel, 379
  sccsdiff, 401
  sort, 427
  style, 444
  test, 481
  troff, 496
  val, 538
  vgrind, 540

r
  ar, 15
  chmod, 63
  dired, 120
  tail, 453
  tar, 456

-raw, stty, 441

-S
  diff, 115
  ld, 238
  make, 278
  refer, 368
  whereis, 553

-s
  cat, 48
  checknr, 58
  comb, 71
  csh, 83
  delta, 108
  diff, 115
  dired, 120
  du, 125
  egrep, 130
  eqn, 144
  error, 145
  fgrep, 165
  finger, 170
  from, 179
  get, 191
  gprof, 199
  graph, 202
  grep, 204
  install, 227
  ld, 238
  ln, 246
  lpr, 259
  lprof, 263
  ls, 266
  mail, 272
  make, 278, 282
  man, 285
  more, 298
  msgs, 301
  neqn, 309
  netstat, 306
  nroff, 316
  page, 323
  pr, 334
  qsub, 351
  refer, 367
  sccsdiff, 401
  sh, 414
  sortbib, 429
  spell, 431
  tc, 460
  test, 481
  tr, 494
  troff, 496
  tty, 505
  val, 538
  w, 543
  what, 549
  whereis, 553

s
  chmod, 64

dired, 120

-s option, dbx, 99

-size, find, 167

-sr option, dbx, 99

-T
ld, 238
lpr, 258
mail, 272
nroff, 316, 384
plot, 330
sort, 428

-t
admin, 4
afio option, 8
autopsy, 25
Concentrix-2800, 25
bfio option, 33
cat, 48
csh, 83
ctags, 94
edit, 128
error, 145
ex, 149
GNU emacs, 133
get, 191
graph, 202
join, 229
ld, 238
lex, 241
lpr, 258
ls, 266
make, 278, 282
man, 285
mon, 294
pr, 334
ruptime, 389
sh, 414
sort, 427
tc, 460
test, 481
troff, 496
ul, 506
vi, 540
ypcat, 580

t
ar, 15
chmod, 64
ps, 342
tar, 456

-t option, date, 97

-tabs, stty, 441

-tandem, stty, 441

-tilde, stty, 443

-tmessage_on, execute, 153

-tostop, stty, 443

-tquiet_exception, execute, 153

-tsignal_on_exception, execute, 153

-type, find, 167

-u
cat, 48
ctags, 94
date, 97
GNU emacs, 134
graph, 202
ld, 238
lint, 244
ls, 266
mail, 272
more, 298
nm, 312
page, 323
ruptime, 389
sh, 414
sort, 428
uniq, 517
whereis, 553

u
ar, 16
chmod, 63
ps, 342
tar, 456

-user
find, 167
GNU emacs, 134

-V
csh, 83
roffbib, 384

-v
cat, 48
csh, 83
ctags, 94
edit, 128
egrep, 130
error, 145
ex, 149
fgrep, 165
ftp, 181
grep, 204
lex, 241
lint, 244
lpr, 259
mail, 272
make, 278
mon, 294
sh, 414
spell, 431
tip, 489
yacc, 578

v
ar, 16
tar, 457

-W, make, 278

-w
adb, 1
ctags, 94
edit, 128
ex, 149
execute, 153
graph, 202
grep, 204
lpr, 258
make, 278
pr, 334
prelf command option, 335
test, 481
troff, 496
vi, 540

w
chmod, 63
dired, 120
tar, 457

-X
csh, 83
ld, 238

-x
col, 68
csh, 83
ctags, 94
ed, 127
edit, 128
ex, 149
fgrep, 165
graph, 202
ld, 238
lint, 244
roffbib, 384
sh, 414
spell, 431
spline, 435

x
ar, 15
chmod, 64
ps, 342
tar, 456

-y
admin, 6
cdc, 54
delta, 109
graph, 202
ld, 238
val, 538

y, mesg, 289

-z
admin, 6
gprof, 200

lint, 244
test, 481

Empty, 17

|, I/O redirection, 84

|&, I/O redirection, 84

||, I/O redirection, 85

=, chmod, 63

<, I/O redirection, 84

>, I/O redirection, 84

>!, I/O redirection, 84

>&, I/O redirection, 84

>&!, I/O redirection, 84

>>, I/O redirection, 84

>>!, I/O redirection, 84

>>&, I/O redirection, 84

>>&!, I/O redirection, 84

\f font change, checknr, 58

\s size change, checknr, 58

~., tip, 489

~!, tip, 490

~?, tip, 490

~#, tip, 490

~^D, tip, 489

~^Z, tip, 490

~|, tip, 490

~<, tip, 490

~>, tip, 490

~c, tip, 489

~p, tip, 489

~s, tip, 489

~t, tip, 490

A

ACC
disable, 152
ELF notes, 152
force use of, 152

ARPANET, mail addresses, 275

%A, addbib, 2

%A%, get, 195

a.out, 1, 200, 237, 312

abscissa, graph, 202

abstract, bibliographic, 2

access
file, find, 167
permission, new file default, 507

access mode, changing, 63

activity
display system, 543
processors, 573

adb, Concentrix-2800 command, 1

add, expr, 159

addbib
and refer, 368
Concentrix-2800 command, 2

address, value replacement, 263

addressing mail, 275

admin
Concentrix-2800 command, 4
SCCS command, 397

affirmative, output repeatedly, 579

afio
manipulate files, 8
options, 8, 33

alias
Concentrix-2800 command, 11
in mail, 273
remove, 509
substitution, 87

all, stty, 440

alloc, Concentrix-2800 command, 12

analysis, writing, 444

analyze, crash dump, Concentrix-2800, 25

apply, Concentrix-2800 command, 13

apropos, Concentrix-2800 command, 14

ar
Concentrix-2800 command, 15
System V format, 16

arbitrary, precision arithmetic
language, 30
library, 30
package, 101

archive
convert to random library, 355
maintenance program, 15
tape, 456

archive names, special case, 9, 34

archiving
afio command, 8
bfio command, 33

argument
echo, 126
delimited by nulls, 197
evaluate
and execute resulting command, 148
as expression, 159
m4, 269
rm, 377

argv
csh, 84
variable, 408

arithmetic
arbitrary-precision
language, 30
package, 101
expression, 269

ARPANET, file transfer, 181

as, Concentrix-2800 command, 17

ASCII, dump format, 318

assembler, Alliant FX/2800, 17

assembly, code, profile, 263

at
Concentrix-2800 command, 19
jobs in queue, 21
removing a job, 22
starting a job, 19

at_sign, Concentrix-2800 command, 23

auto-login, ftp, 181

autopsy, Concentrix-2800 command, 25

awk, Concentrix-2800 command, 26

B

Bourne shell
environment, 419
execution, 420
file name generation, 418
invoking, 414
login, 251
special commands list, 420

Bourne shell commands, continue, 79

Bourne shell options, set command, 411

Bourne shell special command
exec, 151
read, 365
readonly, 366
trap, 495
umask, 508

Bourne shell special commands, times, 488

Bourne special commands, break, 41

Builtin commands, csh list, 91

%B, addbib, 2

%B%, get, 194

background, job, 329
await completion
Bourne shell, 546
C shell, 545
execution, 36
stop, 437

backslash construction, deroff, 111

backspace, col, 68

backward, move, mt, 303

basename Concentrix-2800 command, 29

batch, request
delete, 349
status, 350
submit, 351

baud rate, tip, 489

baudrate, tip, 491

bc, Concentrix-2800 command, 30

beautification, C program, 49

beautify, tip, 491

Bell Laboratories, 2

Benson Varian, 259

Berkeley, interface to SCCS, 397

bfio, manipulate files, 33

bg, Concentrix-2800 command, 36

bibliographic record
display program, 384
sorting, 429

bibliography
create, 2
find reference, 254
refer program, 367

biff Concentrix-2800 command, 37

binary
file
decode from mail, 527
encode from mail, 528
printable strings, 438
install, 227

binary file, locating, 553

binary search, 253

binding, alias, 11

binmail Concentrix-2800 command, 38

blank
diff, 116
expand tab to, 157

blank interpretation, sh, 417

blank line, squeezing, 298, 323

block
amount in file, 446
length, find, 167
number
fgrep, 165
grep, 204
size, dd, 104

bolding, in neqn, 309

Bourne shell
branching, 46
scripts, branching, 46, 218

branch, control structure
multi-way, 448
multiple two-way, 216
simple two-way, 215
unconditional, 198

break
Bourne shell command, 41
Concentrix-2800 command, 40, 41
from switch command, 42

break statement, C, 244

breaksw, Concentrix-2800 command, 42

brk, stty, 442

broadcast, suppression, 289

bs=, dd, 104

bs0, stty, 440

bsf, mt, 303

bsr, mt, 303

buffer, size conversion, 104

bug, report, 406

burst page, suppressing, 259

byte, count, cmp, 67

C

Concentrix-2800 command
addbib, 2
admin, 4
alias, 11
alloc, 12
apply, 13
apropos, 14
ar, 15
at, 19
awk, 26
base name, 29
bc, 30
bg, 36
biff, 37
binmail, 38
break, 40, 41
breaksw, 42
Concentrix-2800 command, 23
get, 190
glob, 197
goto, 198
graph, 202
grep, 204
groups, 206
hashstat, 207
head, 208
help, 209
help - SCCS, 211
history, 212

hostid, 213
hostname, 214
if, 215
if (Bourne shell), 218
if-then-else, 216
indent, 220
indxbib, 225
info, 226
install, 227
jobs, 228
join, 229
keylogin, 232
kill
  Bourne shell, 233
  C shell, 234

Concentrix-2800 command, scc, 393

Concentrix-2800, autopsy, 25

Concentrix-2800 command
adb, 1
afio, 8
as, 17
bfio, 33
emacs, 133
ld, 237
prelf, 335
ptx, 345

%C, addbib, 2

%C%, get, 194

C
extract strings from, 576
extracting error messages, 291
formatting program, 49
program verification, 244
source program, 576
tags file, 94

C language
conditional compilation, 515
formatting source code, 220
ifdef and ifndef, 515
indenting program source, 220

C shell, filenames, 86

caching, NODATACACHE, 153

cal, Concentrix-2800 command, 43

calculate, blocks in file, 446

calculator, desk, 101

calendar
Concentrix-2800 command, 44
display, 43

call graph, gprof, 199

carriage control
Fortran, conversion, 178
Fortran, and lpr, 259

case, mapping with stty, 441

case (Bourne shell), Concentrix-2800 command, 46

case (C shell), Concentrix-2800 command, 45

case command, sh, 415

cat, Concentrix-2800 command, 48

cb, Concentrix-2800 command, 49

cbreak, stty, 440

cbs=, dd, 104

ccat, Concentrix-2800 command, 51

cd, Concentrix-2800 command, 52, 53

cdc, Concentrix-2800 command, 54

cdpath, variable, 408

.errorrc, 145

change
file, group, 61
login, password, 327
login shell, 583
working directory
  cd, 52, 53
  chdir, 56

change full name, ypchfn, 583

changequote, m4, 269

character, count, 548

chdir, Concentrix-2800 command, 56

check, SCCS command, 398

checkeq, Concentrix-2800 command, 57

checknr, Concentrix-2800 command, 58

chfn, Concentrix-2800 command, 327

chgrp, Concentrix-2800 command, 61

child process, variables, 412

chkey, Concentrix-2800 command, 62

chmod, Concentrix-2800 command, 63

chsh, Concentrix-2800 command, 327

cifplot, and lpr, 259

class
file type, 166
scheduling, specification, 152

clean, SCCS command, 398

clear
Concentrix-2800 command, 66
terminal screen, 66

closing delimiter, nroff/troff, 58

cmp, Concentrix-2800 command, 67

code formatting, C, 49

col, Concentrix-2800 command, 68

colcrt, Concentrix-2800 command, 69

colrm, Concentrix-2800 command, 70

column
pr, 334
remove from file, 70

comb, Concentrix-2800 command, 71

comm, Concentrix-2800 command, 73

command
checknr, 58
execute
  in place of current shell, 150, 151
  repetitively, 371
execution
  Concentrix, 430
  remote, 533
expression, csh, 91
find, 167
history, 212
kconfig, 230
pathname, 555
remote execution, 533
rsh, 385
substitution, 89
switch
  break from, 42

default case, 107
label in, 45
vi, 540
xproc, 573

command aliasing, 11

command substitution, sh, 416

commands
Bourne shell, 414
sh, 415

comment
bc, 30
in Ratfor, 356

common block, conversion to data segment, 237

communication, mail, 272

compact, Concentrix-2800 command, 74

comparision, three-way, file, 118

comparison
diff, 115
directory, 116
file, 67, 116
SCCS, 401
sorted, 73

compile
fast, 241
only, bc, 30

compiler
error, analysis, 145
yacc, 578

complex, monitoring, 294

compress
Concentrix-2800 command, 75
file, 51, 74, 75
space to tab, 513

computational complex, monitoring, 294

computing, monitoring resources, 293

computing resource, specification of, 152

comsumption, resource, limit, 242

concatenation, file, 48

Concentrix-2800 command
cal, 43
calendar, 44
case (Bourne shell), 46
case (C shell), 45
cat, 48
cb, 49
ccat, 51
cd, 52, 53
cdc, 54
chdir, 56
checkeq, 57
checknr, 58
chfn, 327
chgrp, 61
chkey, 62
chmod, 63
chsh, 327
clear, 66
cmp, 67
col, 68
colcrt, 69
colrm, 70
comb, 71
comm, 73
compact, 74
compress, 75
continue, 78, 79
cp, 80
crypt, 81
csh, 83
ctags, 94
ctime, 96
date, 97
dbx, 99
dc, 101
dd, 104
default, 107
delta, 108
deroff, 111
df, 112
diction, 114
diff, 115
diff3, 118
dired, 120
dirs, 123
domainname, 124
du, 125
e, 128
echo, 126
ed, 127
edit, 128
egrep, 130
else, 132
end, 140
endif, 141
endsw, 142
enroll, 143
eqn, 144
error, 145
eval, 148
ex, 149
exec, 150, 151
execute, 152
exit, 155, 156
expand, 157
explain, 158
expr, 159, 161
f, 171
false, 163
fg, 164
fgrep, 165
file, 166
find, 167
finger, 170
fmt, 172
fold, 173
for, 174
foreach, 176
fpr, 178
from, 179
fsplit, 180
ftp, 181
last, 235
lastcomm, 236
ld, 237
leave, 240
lex, 241
limit, 242
lint, 244
ln, 246
lock, 247
logger, 248
login, 249, 251
logout, 252
look, 253
lookbib, 254
lorder, 255
lpq, 256
lpr, 258
lprm, 261
lprof, 263
lptest, 265
ls, 266
m4, 269
machid, 271
mail, 272
make, 277, 282
man, 285
mesg, 289
mkdir, 290
mkstr, 291
mon, 293
more, 298
msgs, 301
mt, 303
mv, 305
neqn, 309
netstat, 306
nice
Bourne shell, 310
C shell, 311
nm, 312
nohup
Bourne shell, 313

C shell, 314
notify, 315
nroff, 316
od, 318
on, 321
onintr, 322
page, 323
pagesize, 326
passwd, 327
percent_sign (%), 329
plot, 330
pr, 334
print, 336
printenv, 337
prs, 338
ps, 342
pushd, 347
pwd, 348
qdel, 349
qstat], 350
qsub, 351
quota, 354
ranlib, 355
ratfor, 356
rcp, 358
rdist, 360
refer, 367
rehash, 370
repeat, 371
reset, 372
reten, 373
rev, 374
rlogin, 375
rm, 377
rmail, 378
rmdel, 379
rmdir, 380
roffbib, 384
rpcgen, 381
rsh, 385
rt, 387
rup, 388
ruptime, 389
rusers, 390
rwall, 391
rwho, 392
sccs, 397
sccsdiff, 401
script, 402
sed, 403
sendbug, 406
set, 408, 411
setenv, 412
sh, 414
shift, 422, 423
size, 424
sleep, 425
soelim, 426
sort, 427
sortbib, 429
source, 430
spell, 431
spellin, 433
spellout, 434
spline, 435
split, 436
stop, 437
strings, 438
strip, 439
stty, 440
style, 444
su, 445
sum, 446
suspend, 447
switch, 448
symorder, 450
syscfg, 451
tabs, 452
tail, 453
talk, 454
tar, 456
tbl, 459
tc, 460
tcopy, 461
tcpdump, 462
tee, 475
telnet, 476
test, 481
tftp, 483
time
Bourne shell, 487
C shell, 485
tip, 489
touch, 493
tr, 494
troff, 496
true, 498
tset, 499
tsort, 504
tty, 505
u, 523
ul, 506
umask, 507, 508
unalias, 509
uncompact, 510
uncompress, 511
unexpand, 513
unhash, 514
unifdef, 515
uniq, 517
units, 518
unlimit, 519
unset, 520
unsetnv, 521
uptime, 522
users, 523
uucp, 524
uudecode, 527
uuendecode, 528
uulog, 529
uuname, 530
uusend, 532
uux, 533
vacation, 536
val, 538
vi, 540
vmstat, 542
w, 543
wait
Bourne shell, 546
C shell, 545
wall, 547
wc, 548
what, 549
whatis, 550
whatisin, 551
whereis, 553
which, 555
while (Bourne shell), 558
while (C shell), 556
who, 559
whoami, 560
whois, 561
window, 562
wpwhich, 584
write, 571
xget, 572
xsend, 575
xstr, 576
yacc, 578
yes, 579
ypcat, 580
ypchfn, 583
ypchsh, 583
ypmatch, 582
yppasswd, 583
zcat, 585

concurrency, ld, 238

conditional, in Ratfor, 356

conditional compilatiion, C language, 515

configuration (hardware), 451

confirmation, find, 167

connection
graph line style, 202
to remote system, 489

contents, directory, display, 266

continue
Bourne shell version, 79
Concentrix-2800 command, 78, 79
execution, 78

control
branch
multi-way, 448

multiple two-way, 216
simple two-way, 215
unconditional, 198
test-and-loop control structure, 556, 558

control structure, Ratfor, 356

conv=, dd, 104

conversion
archive to random library, 355
dd file, 104
quantity, 518
underlining for terminal, 506

cooked, stty, 440

copy
binary, 227
dd input files, 104
directory, 80
file, 80, 104
remote, 358, 524
remote file
rcp, 358
uucp, 524
to remote system, 490

copying files, remote hosts, 360

core, dynamic, 12

core file, adb, 1

correspondence, via mail, 272

count
characters, 548
lines, 548
matching, 165, 204
mt, 303
words, 548

count=, dd, 104

cp, Concentrix-2800 command, 80

cpu monitor, xproc, 573

cpu time
display program, 342
monitoring, 294

cr0, stty, 440

crash, dump analyzer, Concentrix-2800, 25

create
directory, 290
hash list for spell, 433
link, 246
tape, 456
user process, 249

crt
previewing, 69
stty, 442

crtbs, stty, 442

crterase, stty, 442

crtkill, stty, 442

crypt, Concentrix-2800 command, 81

csh
and setenv, 412
and stty, 442
Builtin commands, 91
Concentrix-2800 command, 83
exit, 155
list of variables, 408

.cshrc, 83, 84, 299, 324

ctags, Concentrix-2800 command, 94

ctime, Concentrix-2800 command, 96

ctlecho, stty, 442

curses, and mon, 297

curve, interpolation, 435

cwd, variable, 408

D

%D, addbib, 2

%D%, get, 194

daemon, spooling, 258

daily calendar, 44

DARPA Internet users, finding their names, 561

DASI, 316
plotting on, 330

data, page alignment, 237

data type, verification, C, 244

database, relational operator, 229

date
-d option, 97
Concentrix-2800 command, 97
reminder service, 44
set/show, 97

date and time, hexadecimal, 96

dbx
-s option, 99
-sr option, 99
Concentrix-2800 command, 99

dc, Concentrix-2800 command, 101

dd Concentrix-2800 command, 104

dead.letter, 274

debugger, source-level, 99

debugging, mail, 272

dec, stty, 440

decctlq, stty, 442

decimal, dump format, 318

decode
binary file from mail, 527
file, 81

decrement, nice, C shell, 310

decrypt file, 81

decrypt key, keylogin, 232

default
Concentrix-2800 command, 107
swtich command, 107
window size, 540

define statement, Ratfor, 357

definition, m4, 269

delaying command execution, 19

deledit, SCCS command, 398

delete
directory, 377
file, via directory editor, 120

delget, SCCS command, 398

delimiter, error check, 58

delta
Concentrix-2800 command, 108
changing, 108
changing commentary, 54
combining, 71
removal of, 379
SCCS command, 398
sccsdiff comparison, 401

dependency, between files, 277, 282

deroff, and style, 444

deroff Concentrix-2800 command, 111

description, word, 550

desk calculator, 101

df, Concentrix-2800 command, 112

Diablo, 316, 443

diagnostic
  error, compiler, 145
  output file, 269

diagnostics, sh, 420

dialtimeout, tip, 491

diction
  Concentrix-2800 command, 114
  interactive thesaurus for, 158
  wordy or bad sentence, 114

diff Concentrix-2800 command, 115

diff3, Concentrix-2800 command, 118

difference, file, 67, 73
  SCCS versions, 401

Digital Equipment Corp., 188, 441

directory
  change working
    cd, 52, 53
    chdir, 56
  compare, 116
  copy, 80
  create, 290
  dbx, 99
  diff, 115
  display contents, 266
  move, 305
  pathname, 348
  remove, 380
  rename, 305
  rm, 377
  search, 285
  stack, push, 347
  tar, 456
  tree, search for specific file, 167
  unlink, 380
  working, change, 52, 53, 56

dired. *See* emacs

dired Concentrix-2800 command, 120

dirs, Concentrix-2800 command, 123

disable
  five-line header and trailer, 334
  hash table, 514

discard, m4 characters, 269

disconnect, graph, 202

disk
  free space report, 112
  usage summary, 125

disk quotas, 354

disk usage, 354

display
  active job, 228
  bibliographic records, 384
  calendar, 43
  command, pathname, 555
  compressed file, 51
  contents of directory, 266
  directory, 348
    stack, 123
  environment variable, 337
  file
    cat, 48
    pr, 334
  history, 212
  host system
    identifier, 213
    name, 214
  job
    active, 228
    status change, 315
  last command, 236
  line, first, 208
  mail, header line, 179
  memory usage, 230
  name, current domain system, 124
  network status, 306
  object program size, 424
  page size, 326
  process statistics, 230
  processes, 573
  processor activity, 573
  remote status, 389
  system, activity, 543
  system configuration, 230
  text file, 298, 323
  uptime, 522
  user
    current, 523
    id, 560
    remote, 391

display file, in dump format, 318

display status, on processes, 342

display-oriented editor
  GNU emacs, 133
  vi, 540

ditroff, and lpr, 258

diversion, m4, 269

division, expr, 159

divnum, m4, 269

dnl, m4, 269

document, style analysis, 444

documentation
  hierarchical reader, 226
  locating, 553
  on-line, 285
    contents, 551

dollar sign character, cat, 48

domain, system name, 124

domainname
  and ypcat, 580
  Concentrix-2800 command, 124

draw, graph, 202

dseek=, dd, 104

dst, date, 97

dsusp, stty, 443

du, Concentrix-2800 command, 125

dump, 11
  crash analyzer, Concentrix-2800, 25
  ELF file as text, 335
  traffic on network, 462

dump program, 318

dumpdef, m4, 269

DVI, and lpr, 258

dynamic, core, 12

E

ELF file, dump as text, 335

ELF note, execute command, 152

ELF notes, ACC, 152

Exabyte, tar limitation, 456

%E, addbib, 2

%E%, get, 194

e, Concentrix-2800 command, 128

ebcdic, 188

echo
  argument, 126
    delimited by nulls, 197
  Concentrix-2800 command, 126
  controlling with stty, 440
  csh, 83
  stty, 440
  variable, 408

echocheck, tip, 491

ed
  and diff3, 118
  Concentrix-2800 command, 127
  script, diff, 115

edit
  Concentrix-2800 command, 128
  SCCS command, 398
  startup commands, 412

editor
  choice of in mail, 274
  dired, 120
  ed, 127
  ex, 149
  GNU emacs, 133
  simple, 128
  stream (non-interactive), 403
  vi, 540

egrep, Concentrix-2800 command, 130

ek, stty, 440

else, Concentrix-2800 command, 132

emacs, Concentrix-2800 command, 133

enable, interrupt handling, 322

encode
  binary file from mail, 528
  file, 81

encrypt
  file, 81
    ed, 127
  mail
    password, 143
    read, 572
    send, 575

encryption
  and edit, 128
  and ex, 149
  change key, 62

end
  Concentrix-2800 command, 140
  file, 303

endif, Concentrix-2800 command, 141

endnote, finding and formatting, 367

endsw, Concentrix-2800 command, 142

enroll Concentrix-2800 command, 143

entry point, specifying, 237

environment
  sh, 419
  variable
    display, 337
    remove, 521

environment variable
  LDPATH, 238
  MORE, 299, 324
  PRINTER, 258
  REFER, 367
  setting, 412

eof
  mt, 303
  stty, 442

eofread, tip, 491

eofwrite, tip, 491

eol, tip, 491

eqn
  and checkeq, 57
  and refer, 368
  Concentrix-2800 command, 144
  remove construct, 111

equation, in neqn, 309

equation delimiter, eqn, 144

erase, stty, 442

error
  Concentrix-2800 command, 145
  check
    nroff delimiter, 58
    spelling, 431
  diagnostic, compiler, 145
  message
    fgrep, 165
    grep, 204
  reporting, chgrp, 61

error message, extracting from C code, 291

errprint, m4, 269

escape, tip, 491

etxack, stty, 443

eval
  Concentrix-2800 command, 148
  m4, 269

evaluate, argument
  as expression, 159
  execute resulting command, 148

even, stty, 440

event, history substitution, 85

everything, stty, 440

ex
  Concentrix-2800 command, 149
  statup commands, 412
  tags file, 94

exception, execute, 153

exceptions, tip, 491

exec, Concentrix-2800 command, 150, 151

execute
  -tmessage_on, 153
  Concentrix-2800 command, , 152
  command
    find, 167
    from file, 430
    in place of current shell, 150, 151
    repetitively, 371
    vi, 540
  disable ACC, 152
  foreground job, 164
  immediately in dbx, 99
  m4, 269
  modifiy header, 153
  on a remote system, 321
  remote, 533
  suspend, 425
  use ACC, 152

execute -ce, 154

execute -w, write options to header, 154

execute
  background job, 36
  command, later, 19

execution
  continue, 78
  continue (Bourne version), 79

on detached CE, 238
on multiple arguments, 13
program, on specific resource type, 152
resume, after foreach or while, 40
sh, 420
timing, 154, 485, 487

execution profile, 199

execven, and setenv, 412

EXINIT, environment variable, 412

exit
Concentrix-2800 command, 155, 156
csh, 155
from for or while, 41
sh, 156
status
non-zero, 163
zero, 167
system call, 201, 245

expand
Concentrix-2800 command, 157
string search, 438
tab to space, 157

expand a file, 511, 585

explain
Concentrix-2800 command, 158
ed output, 127

expr, Concentrix-2800 command, 159, 161

expression
command, csh, 91
evaluate, argument as, 159
evaluation, 481
grep, 204
numeric, csh, 89
regular, searching, 130
relational, Ratfor, 357

external variable, and ld, 238

extract
file in tar, 456
strings from C program, 576

F

%F, addbib, 2

%F%, get, 194

f, Concentrix-2800 command, 171

false, Concentrix-2800 command, 163

fast search for pattern, 165

fcnname, scc command option, 394

ff0, stty, 441

ffline, mt, 303

fg, Concentrix-2800 command, 164

fgrep, Concentrix-2800 command, 165

field
m, join, 229
separator, awk, 26

file
access
find, 167
permissions, 507, 508
binary, 227
decode from mail, 527
encode from mail, 528
printable strings in, 438
blocks in, 446
Concentrix-2800 command, 166
column removal, 70
compare, 67, 73, 116
comparison, three-way, 118
compress, 74, 75
compressed dump, 25
concatenate, 48
contents, m4, 269
convert, 104
copy, 80, 104
count, 303
creation, mask, 507, 508
date modified, 493
decode, 81
decrypt, 81
deletion, via directory editor, 120
display, 334
compressed, 51
first lines, 208
last part, 453
more, 298
page, 323
simultaneous, 334
text, 48
display name, 266
encode, 81
encrypt, 81
entry in du, 125
executable, 237
execution, command from, 430
existence, 481
find, 167
format, 334
GNU emacs, 133
inaccessible, 269
merging, 427
mode, uusend, 532
modify, find, 167
move, 305
permission, changing, 63
print, 336
printing, 258
remote
copy, 358, 524
send, 532
remove, interactively, 377
rename, 305
repeated line in, 517
reverse line in, 374
search
directory tree, 167
pattern, 204
fast, 165
segment size in sccsdiff, 401
split, 436
tags, 94
uncompress, 510
unlink, 377
updating, 277, 282

file compression, 75

file copying, remote machines, 360

file distribution, multiple machines, 360

file expansion (after compression), 511, 585

file mode, display, 267

file name, listing, 266

file transfer, ARPANET, 181

file transfers, remote machines, 483

filename
find, 167
sh, 418
substitution, 89
testing, 90

filenames
C shell, 86
completion, 86
partial, 86

files, uncompression, 511, 585

files=, dd, 104

filter, reverse line feed, 68

find
bibliographic reference, 254
Concentrix-2800 command, 167
ordering relation in object library, 255
printable strings in binary file, 438
spelling error, 431

finger
Concentrix-2800 command, 170
lookup, 170

finger command, changing the information, 327

first
lines of file, 208
n characters and fields, uniq, 517

fix, SCCS command, 399

flow control, terminal, 441

flow of control, in Ratfor, 356

flush, stty, 443

flusho, stty, 443

fmt Concentrix-2800 command, 172

fold
Concentrix-2800 command, 173
lines, long for narrow output device, 173

font, eqn, 144

footnote, 2
finding and formatting, 367

for, Concentrix-2800 command, 174

for command, sh, 415

force, tip, 491

force 132 column output, prelf command, 335

force 80 column output, prelf command, 335

foreach
Concentrix-2800 command, 176
resume execution after, 40

foreground job
fg, 164
percent_sign, 329

form feed, pr, 334

format
file, pr, 334
text
simple, 172
troff, 497

formatting
code, C, 49
mathematics, 309
text, nroff, 316

formfeed, suppressing, 298, 323

Fortran
carriage control
and lpr, 259
conversion, 178
compiler, 173
preprocessor, Ratfor, 356
split up multi-routine file, 180
tags file, 94

forward, move, 303

fpr, Concentrix-2800 command, 178

framesize, tip, 491

free, disk space, 112

from, Concentrix-2800 command, 179

fsf, mt, 303

fsplit Concentrix-2800 command, 180

fsr, mt, 303

ftp, Concentrix-2800 command, 181

function, bc, 31

FX/Fortran, and error program, 145

G

GNU Emacs, -kill, 133

GNU emacs, X window system, 135

%G, addbib, 2

%G%
get, 194
SCCS command, 400

GCOS, C compatiblity checking, 244

General Electric, 316, 441

generator, lexical analysis programs, 241

get
Concentrix-2800 command, 190
SCCS command, 398

glob, Concentrix-2800 command, 197

gmon.sum, 199

GMT. *See* Greenwich Mean Time

gname file, find, 167

goto, Concentrix-2800 command, 198

government, order number, 2

gprof, Concentrix-2800 command, 199

graph
and spline, 435
Concentrix-2800 command, 202
computing resources, 294
draw, 202

Greenwich Mean Time, 97

grep, Concentrix-2800 command, 204

group
change, 61
id, change, 61
install, 227
membership, 206

grouping, statement, in Ratfor, 356

groups Concentrix-2800 command, 206

GSI, plotting on, 330

H

%H, addbib, 2

%H%, get, 194

half-hearted, diff, 116

half-line
characters, 69
col, 68
colcrt, 69

hangup, signal
Bourne shell, 313
C shell, 314
ignore, 313, 314

hard, link, 246

hardware configuration, 451

hash
  list
    spell, 433
    spellout, 434
  table, 370
    disable, 514
    statistics, 207

hashstat, Concentrix-2800 command, 207

head, Concentrix-2800 command, 208

header
  line, 179
  w, 543

height, graph, 202

help, Concentrix-2800 command, 209

help - SCCS, Concentrix-2800 command, 211

hexadecimal, dump format, 318

histchars, variable, 409

history
  Concentrix-2800 command, 212
  source, 430
  spell, 431
  substitution, 85
  variable, 409

HOME
  environment variable, 412
  shell variable, 414
  su, 445
  tip, 492

home, variable, 409

host
  identifier, 213
  name, 214
  remote, 532
  system, 213, 214
    status, 389
  tip, 491

hostid Concentrix-2800 command, 213

hostname Concentrix-2800 command, 214

Huffman code, compress file, 74

I

%I, addbib, 2

%I%
  get, 194
  SCCS command, 400

IBM, C compatibility checking, 244

ibs=, dd, 104

id
  group, change, 61
  process, m4, 269
  user, 560

identification
  number, jobs, 228
  SCCS file, 549

if
  Concentrix-2800 command, 215
  sh, 415

if (Bourne shell), Concentrix-2800 command, 218

if-then-else, Concentrix-2800 command, 216

if=, dd, 104

ifdef, m4, 269

ifdef and ifndef, C language, 515

ifelse, m4, 269

.IGNORE, make, 277, 282

ignore
  hangup signal
    Bourne shell, 313
    C shell, 314
  interrupts, 322

ignoreeof, variable, 409

IMP, host table, 306

include
  file in nroff input, 426
  m4, 269

include files, add to search list, 17

include statement, Ratfor, 357

incr, m4, 269

increment, nice, 310

incremental loading, 237

indent, Concentrix-2800 command, 220

indenting, C program source, 220

index
  create, 345
  ctags, 94
  inverted for bibliography
    build, 225
    find reference, 254
  m4, 269

indxbib, and refer, 368

indxbib Concentrix-2800 command, 225

info
  Concentrix-2800 command, 226
  SCCS command, 399

information, lpq file, 256

inode, number, 167

input
  block, size, 104
  file name, 104
  skip n records, 104
  uux, 533
  xstr, 576

input/output, sh, 418

install
  binary file, 227
  Concentrix-2800 command, 227

interactive, copy, 80

interface, netstat, 306

Internet, address, 476

Internet TFTP, 483

Internet users, finding their names, 561

interpolation, curve, 435

interpretation, mv argument, 305

interrupt
  handling, 322
  ignore, 322

intr, stty, 442

i-number, display, 266

inv, indxbib, 225

inverted bibliographic index
  build, 225
  find reference, 254

I/O redirection, 84

iseek=, dd, 104

iterated loop construct, 174, 176

J

%J, addbib, 2

job
  active, 228
  background
    await completion
      Bourne shell, 546
      C shell, 545
    execute, 36
    percent_sign, 329
    stop, 437
  foreground, 164, 329
  number, 164
  status change, 315
  termination, 234

jobs, Concentrix-2800 command, 228

join, Concentrix-2800 command, 229

K

%K, addbib, 2

kconfig, command, 230

key, encryption, 62

key map, yellow pages, 582

keylogin, Concentrix-2800 command, 232

kill
  Concentrix-2800 command
    Bourne shell, 233
    C shell, 234
  running batch requests, 349
  stty, 442

kilobytes, grand total in du, 125

L

%L, addbib, 2

%L%, get, 194

label, switch command, 45

language, error, analysis, 145

last, Concentrix-2800 command, 235

lastcomm, Concentrix-2800 command, 236

lcase, stty, 441

ld
  Concentrix-2800 command, 237
  scc, 393

LDPATH, environment variable, 238

leading, history number, 212

leave
  Concentrix-2800 command, 240
  reminder, 240

Lempel-Ziv coding
  compress file, 75
  uncompress file, 511, 585

len, m4, 269

letter case
  fgrep, 165
  grep, 204

level, line profiler, 263

lex, 244
  Concentrix-2800 command, 241

lexical, analysis, 241

lib file, adb, 1

library
  and linkage editor, 237
  convert to random, 355
  maintenance program, 15
  ordering relation, 255, 504

limit
  Concentrix-2800 command, 242
  resource consumption, 242

line
  count, 548
  feed, reverse, 68
  level profiler, 263
  number
    cat, 48
    fgrep, 165
    grep, 204
  printer
    examine queue, 256
    remove job, 261
  repeated, 517
  reverse, 374
  split, 436

line printer
  default, 412
  spooling command, 258

link
  create, 246
  symbolic
    and C shell, 91
    used by lpr, 259

link editor, scc, 393

linkage editor, ld, Concentrix-2800, 237

lint, Concentrix-2800 command, 244

Lisp, editing, 128, 149

list
  active job, 228
  file, 456
  join, 229
  name, rearrange, 450
  sh, 415
  system users, 523
  UUCP host names, 530

listing, assembly, 17

litout, stty, 443

ln, Concentrix-2800 command, 246

load
  dump into memory, 25
  time sort, 389

load map, 238

loader, ld, 237

loading, incremental, 237

local, numeric variable, 23

lock, Concentrix-2800 command, 247

logger, 248

.login, 84, 301

login
  Bourne shell special command, 251
  Concentrix-2800 command, 249, 251
  display, all local machine users, 392
  password, 327
  record of, 235
  remote, 375
  shell, terminate, 252

login directory, 412

login name, show, 559
login shell, change, 583
logins, network, 390
.logout, 84
logout
  Concentrix-2800 command, 252
  record of, 235
long
  format
    diff, 115
    dirs, 123
    finger, 170
  line, narrow output device, 173
  output, 543
look, Concentrix-2800 command, 253
lookbib Concentrix-2800 command, 254
lookup program, user information, 170
loop
  in Ratfor, 356
  iterated loop construct, 174, 176
lorder, and ar, 16
lorder Concentrix-2800 command, 255
lpq Concentrix-2800 command, 256
lpr, Concentrix-2800 command, 258
lprm Concentrix-2800 command, 261
lprof, Concentrix-2800 command, 263
ls
  Concentrix-2800 command, 266
  examining access modes, 63

M

Model 200 processor
  disable ACC, 152
  force use of ACC, 152
%M, addbib, 2
%M%
  get, 194
  SCCS command, 400
m4, Concentrix-2800 command, 269
machid, Concentrix-2800 command, 271
machine id, machid command, 271
macro
  bibliographic, 384
  checknr, 58
  m4 processor, 269
  nroff, 316
  remove definition, 111
magnetic tape
  mt, 303
  read, 387
magnetic tapes
  format of unknown tapes, 461
  making a copy, 461
mail
  and lpr, 259
  arrival notification, 37
  binmail, 38
  Concentrix-2800 command, 272
  display, header line, 179
  encrypted
    password, 143
    read, 572
    send, 575
  remote, 378
  to msgs program, 302
  uucp, 524
  variable, 409
  while you are on vacation, 536
mail addressing, description, 275
Mail.rc, 272
maintain, uucp/uux transaction log, 529
maintenance, of programs, 277, 282
make
  Concentrix-2800 command, 277, 282
  directory, 290
  link, 246
  using with error program, 145
makefile, 277, 282
maketemp, m4, 269
man, locating, 553
man Concentrix-2800 command, 285
manipulate files
  afio, 8
  bfio, 33
map master, yellow pages, 584
match
  finger arguments, 170
  lines, fgrep, 165
mathematics
  error checking, 57
  formatting, 309
  preprocessor, troff, 144
mdmbuf, stty, 443
me macro package, 58
memory
  load dump into, 25
  management routine, 306
memory usage, display, 230
merge
  diff, 116
  SCCS, 398
merging, of files, 427
mesg, Concentrix-2800 command, 289
message
  controlling, 289
  sending via mail, 272
  system, reading, 301
metacharacter, csh, 89
misspelling, stop list, 431
mkdir, Concentrix-2800 command, 290
mkey, indxabib, 225
mkstr, Concentrix-2800 command, 291
mode, access, changing, 63
Model 33, Teletype, 441
Model 37, Teletype, 441
mon
  and curses, 297
  Concentrix-2800 command, 293
  thread_id, 297
monitor, system resources, 293
MORE, environment variable, 299, 324

more
  and msgs, 301
  Concentrix-2800 command, 298

move
  directory, 305
  file, 305

ms, and refer, 368

ms macro package, 58

msgs, Concentrix-2800 command, 301

.msgsrc, 301

mt Concentrix-2800 command, 303

mtime, find, 167

multi-column output, col, 68

multi-routine Fortran file, split up, 180

multiplication, expr, 159

mv Concentrix-2800 command, 305

N

NIS
  data base, display, 580
  password, changing, 583

%N, addbib, 2

name
  base, strip to, 29
  current domain system, 124
  host, 214
  list, rearrange, 450
  list (symbol table), 450
  local system, 530
  m4, 269
  user, write, 571

name list, display program, 312

named, file, add, 456

names, UUCP hosts, 530

narrow output device, fold long line for, 173

neqn
  and checkeq, 57
  and refer, 368
  Concentrix-2800 command, 309

.netrc, 181

netstat, Concentrix-2800 command, 306

network
  address, 306
  dump traffic, 462
  logins, 390
  status, 306
  users, 392

network security, keylogin, 232

new, stty, 443

newline
  echo, 126
  fgrep, 165
  grep, 204
  m4, 269

nice, Concentrix-2800 command
  Bourne shell, 310
  C shell, 311

nl, stty, 441

nl0, stty, 441

nm, Concentrix-2800 command, 312

noclobber, 84
  variable, 409

noglob, variable, 409

nohang, stty, 443

nohup, Concentrix-2800 command
  Bourne shell, 313
  C shell, 314

non-matching, lines
  fgrep, 165
  grep, 204

non-printing, character, 48

nonomatch, variable, 409

notification
  batch request completion, 351
  mail arrival, 37

notify
  Concentrix-2800 command, 315
  variable, 409

nroff
  and more, 298
  and neqn, 309
  and page, 323
  and refer, 367
  and roffbib, 384
  and sortbib, 429
  Concentrix-2800 command, 316
  delimiter error check, 58
  eliminating .so, 426
  remove constructs, 111
  table format, 459

nulls, echo arguments delimited by, 197

number, characters, 269

numeric, local variable, 23

O

Object files, Scalar/C, 393

Optimization
  Scalar/C, 393
  scc, 393

%O, addbib, 3

object, program
  linking, 237
  size, 424

object file
  adb, 1
  assembler, 17

octal, dump format, 318

od, Concentrix-2800 command, 318

odd, stty, 441

of=, dd, 104

offset, strings, 438

on, Concentrix-2800 command, 321

on-line documentation, 285
  contents, 551
  finding, 14
  locating, 553

onintr, Concentrix-2800 command, 322

opening delimiter, nroff/troff, 58

operand
  bc, 30
  file, csh, 90
  numeric, csh, 90

operator
  bc, 30
  csh, 89
  relational database, 229

options for X window, GNU emacs, 135

ordering, relation, object library, 255

original mode, tar, 456

originating account, rsh, 385

oseek=, dd, 104

output
block, size, 104
diagnostic, 269
file name, dd, 104
join, 229
lex, 241
pipe through cat -s, 285
sccsdiff, 401
send to troff, 285
stream, 269
uniq, 517

output dependency list
as command, 17
assembler, 17

owner, install, 227

P

%P, addbib, 3

%P%, get, 194

page
Concentrix-2800 command, 323
header, pr, 334
length, 334
size, 326
width, pr, 334

pagesize Concentrix-2800 command, 326

parameter substitution, sh, 416

parity
in dump format, 318
in terminal I/O, 440

parsing
table, 578
yacc, 578

Pascal, tags file, 94

/etc/passwd, 428, 580

passwd, Concentrix-2800 command, 327

password
change, 327
encrypted mail, 143
secret key, 62, 232
yellow pages, changing, 583

PATH, environment variable, 412

path
search, 408
table, recompute, 370
variable, 409

pathname
command, 555
find, 167
modification, 88

pattern
awk, 26
file, 26
scanning, 26
search file for, 165, 204
searching, 130
what, 549

pendin, stty, 443

per-protocol statistics, netstat, 306

percent_sign (%), Concentrix-2800 command, 329

permission
access, new file default, 507, 508
file, changing, 63
find, 167

permuted index, ptx command, 345

phone program, 454

phototypesetter
roffbib, 384
simulator, 460
troff, 496

pid
See also process identification number
monitoring, 293

pipe, 84
output through cat -s, 285

pipeline, 84
sh, 415

.plan file, finger, 170

plot
and lpr, 259
Concentrix-2800 command, 330
graph, 202

plotting, on terminal, 330

point
size, eqn, 144
size, troff, 497

positional parameters, shift (Bourne shell), 423

pr
and lpr, 258
Concentrix-2800 command, 334
scssdiff output, 401

prefix, strip, 29

prelf, Concentrix-2800 command, 335

preprocessor
mathematics, neqn, 309
Ratfor (Rational Fortran), 356
Scalar/C, 393

print
Concentrix-2800 command, 336
file, 336
hash table statistics, 207
record size, 387
SCCS file, 338

printenv, Concentrix-2800 command, 337

PRINTER, environment variable, 258, 412

printer
line, default, 412
lpq, 256

printers, testing them, 265

printing, files, 258

priority
batch request, 351
display program, 343
set
Bourne shell, 310
C shell, 311

process
background, await completion
Bourne shell, 546
C shell, 545
display last, 236
termination, 233, 234
user, create new, 249

process statistics, kconfig, 230

process status, display program, 342

process time, display, 573

processor, force no caching, 153

processors, activity, 573

prof, and gprof, 200

.profile, 299, 301, 324, 414

profile, execution, 199

program
  building, 277, 282
  execution, on specific resource type, 152
  maintenance, 277, 282

program verification, C, 244

prompt
  mv, 305
  tip, 491
  variable, 409

prompting, sh, 418

protection, file. *See* chmod

proto, netstat, 306

protocol
  block address, control, 306
  file transfer, ARPANET, 181

prs
  Concentrix-2800 command, 338
  SCCS command, 399

ps
  Concentrix-2800 command, 342
  thread_id, 344

ptx, Concentrix-2800 command, 345

push, directory stack, 347

pushd, Concentrix-2800 command, 347

pwd, Concentrix-2800 command, 348

Q

%Q, addbib, 3

%Q%
  admin, 5
  get, 194

qdel, Concentrix-2800 command, 349

qstat, Concentrix-2800 command, 350

qsub, Concentrix-2800 command, 351

quantitative conversion program, 518

queue
  batch request, 351
  examine, 256
  lprm printer, 261
  remove, job, 261

queues
  displaying list of jobs, 21
  removing a job, 22

quit, stty, 442

quota, Concentrix-2800 command, 354

quotation
  csh, 87
  sh, 418

R

RPC Language, 381

%R, addbib, 3

%R%
  get, 194
  SCCS command, 400

raise, tip, 492

raisechar, tip, 492

ranlib
  and ar, 16
  and ld, 239
  System V, 355

ranlib Concentrix-2800 command, 355

ratfor, Concentrix-2800 command, 356

Rational Fortran, 356

raw, stty, 441

rcp Concentrix-2800 command, 358

rdist, Concentrix-2800 command, 360

read
  Bourne shell special command, 365
  encrypted mail, 572
  m4, 269
  magnetic tape, 387
  on-line documentation, 285

read crash dump, autopsy, Concentrix-2800, 25

readability, analysis, 444

reader, hierarchical documentation, 226

readnews, 272

readonly, Bourne shell special command, 366

recompute, path table, 370

record
  mt, 303
  size, 387
  tip, 492

recovery, after crash, vi, 540

recursive
  copy, 80
  diff, 115

redirect, batch request standard output, 351

redirection, i/o, 84

REFER, environment variable, 367

refer, 2
  and sortbib, 429
  Concentrix-2800 command, 367

reference, bibliographic, 254

regular expression
  expr, 159
  searching, 130

rehash, Concentrix-2800 command, 370

relational, join, 229

relational expression, Ratfor, 357

relative, line number, 165

relocation
  and ld, 238
  bits, remove, 439

reminder
  date, 44
  leaving time, 240

/etc/remote, 489

remote
  file
    rcp, 358
    send, 532
  login, 375
  mail, 378
    uucp, 378

shell, 385
status, 389
user, 375
users, write to all, 391

remote connection, via tip, 489

remote hosts, 360

remote machines
file transfers, 483
uptime, 388

remote procedure calls, 381

remote systems, executing a command on a remote sytsem, 321

remove
alias, 509
column from file, 70
directory, 380
environment variable, 521
file, rm, 377
force, 377
line, printer job, 261
local variable, 520
nroff/troff constructs, 111
relocation bits, 439
resource limitation, 519
symbol table, 439

rename
directory, 305
file, 305

repeat
Concentrix-2800 command, 371
command execution, 371

repeated line; uniq, 517

replacement, empty output field, join, 229

report
disk free space, 112
hash list for spell, 434
repeated line, 517

request
active, 350
delayed, 350
long format listing, 350
run after specified time, 351

reserve, terminal, 247

reset
Concentrix-2800 command, 372
teletype bits, 372

resource
computing, specification of, 152
consumption, limit, 242
limitation, remove, 519

resources, monitoring, 293

resume
execution, after foreach or while, 40
job in background, 329

reten Concentrix-2800 command, 373

retension, tape cartridge, 373

return, exit status, value nonzero, 163

Return statement, Ratfor, 357

rev, Concentrix-2800 command, 374

reverse
display of commands, 236
history command, 212
line, 374

rewind, mt, 303

rewoffl, mt, 303

rlogin, Concentrix-2800 command, 375

rm, Concentrix-2800 command, 377

rmail Concentrix-2800 command, 378

rmdel, Concentrix-2800 command, 379

rmdir, Concentrix-2800 command, 380

roffbib, 3
and refer, 368
Concentrix-2800 command, 384

routine, profile specified, 263

routing, table, 306

RPC protocols, 381

rpcgen, Concentrix-2800 command, 381

rprnt, stty, 443

rsh Concentrix-2800 command, 385

rt Concentrix-2800 command, 387

runoff. *See* nroff, troff

rup, Concentrix-2800 command, 388

ruptime Concentrix-2800 command, 389

rusers, Concentrix-2800 command, 390

rwall, Concentrix-2800 command, 391

rwho Concentrix-2800 command, 392

S

Scalar/C
object files, 393
optimization, 393
scc command, 393

Scalar/C compiler, scc1, 393

Scalar/C compiler command, scc, 393

Scalar/C preprocessor, scc0, 393

Setting signal, Bourne shell trap command, 495

Special commands, Bourne Shell, 420

System V
ar, 16
ranlib, 355

%S, addbib, 3

%S%, get, 194

same report, diff, 115

save, graph screen, 202

savehist, variable, 409

scanning, pattern, 26

scc
Concentrix-2800 command, 393
optimization, 393

scc0, Scalar/C preprocessor, 393

scc1, Scalar/C compiler, 393

SCCS
See also Source Code Control System
administer s-files, 397
administration, 4
Berkeley interface, 397
change delta, 108
changing delta commentary, 54
combine deltas, 71

command, 211
delta, 398
error, 211
file
compare, 401
identify, 549
file creation, 4
file printing, 338
file validation, 538
get command, 190
get file for compilation, 398
get file for editing, 398
list edited files, 399
merge file, 398
print id keywords, 399
removal of delta, 379
remove file, 399
report changes, 399

sccs, Concentrix-2800 command, 397

sccsdiff Concentrix-2800 command, 401

scheduling, monitoring of, 296

scheduling class, specification, 152

scheduling increment, display program, 343

screen, GNU emacs editor, 133

screen control, windows, 562

script
Concentrix-2800 command, 402
diff, 116
tip, 492

script editor, 403

scripts, Bourne shell, branching, 218

search
binary, 253
directory, 285
tree, 167
file, pattern, 204
fast, 165
regular expression, 130
string, 253

search path
prefix setting, 412
setting, 408

secure network, keylogin, 232

sed, Concentrix-2800 command, 403

seek=, dd, 104

send
encrypted mail, 575
file to remote host, 532
from, 179
mail, 378

sendbug, Concentrix-2800 command, 406

sendmail, 272

sentence, wordy or bad, 114

server, yellow pages, 584

set
Bourne shell options, 411
Concentrix-2800 command, 408, 411
file, creation mask, 507, 508
host, identifier, 213
local, C shell variable, 408
name, current domain system, 124
priority
Bourne shell, 310
C shell, 311
variable, local numeric, 23

setenv, 299, 324
Concentrix-2800 command, 412

setting X window options, GNU emacs, 135

sh, 416
blank interpretation, 417
Concentrix-2800 command, 414
case command, 415
command substitution, 416
diagnostics, 420
environment, 419
execution, 420
exit, 156
filename generation, 418
for command, 415
if command, 415
input/output, 418
list, 415
parameter substitution, 416
pipeline, 415
prompting, 418
quotation, 418
signals, 419
simple command, 414
while, 415

SHELL
environment variable, 412
su, 445
tip, 492

shell
Bourne, 414
choice of in mail, 274
default login, 412
execute commands in place of current, 150, 151
interpret commandfile with specified, 351
login, 252
remote, 385
stop current, 447
variable, 409

shells, changing the default, 327

shift
Concentrix-2800 command, 422, 423
variable members, 422

short
format, 170
output, 543

SIGINT, and sh, 414

signal, send to process, 233, 234

signals, sh, 419

SIGQUIT, and sh, 414

SIGTERM
and sh, 414
lastccomm, 236

.SILENT, make, 278, 282

simple command, sh, 414

sinclude, m4, 269

single, spacing, 48

size, Concentrix-2800 command, 424

skip=, dd, 104

sleep, Concentrix-2800 command, 425

.so, nroff, elimination, 426

socket
display, 267
state, 306

soelim, Concentrix-2800 command, 426

sort
Concentrix-2800 command, 427
file, compare, 73
topological, 504
users, 389

sortbib, 384
Concentrix-2800 command, 429

sorting, bibliographic records, 429
source
  Concentrix-2800 command, 430
  debugger, 99
  file, rcp, 358
Source Code Control System. *See* SCCS
source file, locating, 553
space
  compress to tab, 513
  expand tab to, 157
  report free disk, 112
spacing
  graph, 202
  tab, 68
special file, display, 266
specify symbol, as command, 17
spell
  British, 431
  Concentrix-2800 command, 431
  error, 431
  hash list
    create new, 433
    report on, 434
spellin Concentrix-2800 command, 433
spellout Concentrix-2800 command, 434
spline, Concentrix-2800 command, 435
split
  Concentrix-2800 command, 436
  Fortran file into routines, 180
  file, 436
spooling
  daemon, 258
  queue, 256
stack, directory
  display, 123
  push, 347
standard input
  redirection, 84
  splitting, 475
standard output, redirection, 84
Stanford, 258
start, stty, 442
statement, bc, 31
statistics
  hash table, 207
  lex, 241
  virtual memory, 542
status
  host system, 389
  job, 315
  mt, 303
  network, 306
  system resources, 293
  variable, 409
stop
  background job, 437
  Concentrix-2800 command, 437
  misspelling list, 431
  shell, current, 447
  stty, 442
string
  argument, 159
  list, 165
  searching for, 253
strings
  Concentrix-2800 command, 438
  extract C program, 576
  printable in binary file, 438
strip
  binary in install, 227
  Concentrix-2800 command, 439
  name to base, 29
stripping, of output files, 238
stty, Concentrix-2800 command, 440
style, Concentrix-2800 command, 444
su, Concentrix-2800 command, 445
subexpression, expr, 159
subprogram, split, 180
subscript, in neqn, 309
substitution
  alias, 87
  command, 89
  filename, 89
  history, 85
  of userid, 445
  variable, 87
substr, m4, 269
substring, m4, 269
subtraction, expr, 159
suffix, strip, 29
.SUFFIXES, make, 278, 282
sum, Concentrix-2800 command, 446
summary, disk usage, 125
superscript, in neqn, 309
superuser, becoming, 445
suppress
  newline echo, 126
  statistics, 241
surface, characteristic, analysis, 444
susp, stty, 443
suspend
  Concentrix-2800 command, 447
  execution, 425
switch
  Concentrix-2800 command, 448
  command
    break from, 42
    default case, 107
    label in, 45
symbol
  resolving undefined, 237
  table, remove, 439
symbol table, display program, 312
symbolic, link, ln, 246
symbolic link
  and C shell, 91
  display, 266
  used by lpr, 259
symbolic name, for command, 11
symlink, system call, 259
symorder Concentrix-2800 command, 450
synopsis, on-line documentation, 285
syscfg, Concentrix-2800 command, 451
syscmd, m4, 269
system
  activity, 543
  local system name, 530

message, reading, 301
system configuration, 451
system log, making entries, 248
system resources, monitoring, 293

T

%T, addbib, 3
%T%, get, 194
tab
  character
    cat, 48
    join, 229
    replacing, 441
  compress space to, 513
  expand to space, 157
  set, 452
  unexpand, 513
tab0, stty, 441
tabexpand, tip, 492
table
  descriptor, remove, 111
  format for nroff/troff, 459
  hash
    disable, 514
    statistics, 207
tabs
  Concentrix-2800 command, 452
  stty, 441
tag
  edit, 128
  ex, 149
  vi, 540
tags file, creating, 94
tail, Concentrix-2800 command, 453
talk
  and mesg, 289
  Concentrix-2800 command, 454
tandem, stty, 441
tape
  archive, 456
  cartridge retension, 373
  manipulate, 303
tar, Concentrix-2800 command, 456
tbl
  and refer, 368
  Concentrix-2800 command, 459
tc, Concentrix-2800 command, 460
tcopy, Concentrix-2800 command, 461
tcpdump, Concentrix-2800 command, 462
tee, Concentrix-2800 command, 475
tek, stty, 441
Tektronix, 441
  plotting on, 330
telephone program, 454
teletype bits, reset, 372
Teletype Corp., 316, 441
tell, SCCS command, 399
TELNET, interface, 476
telnet, Concentrix-2800 command, 476
TERM, environment variable, 412
term, variable, 409
TERMCAP, environment variable, 412
termcap, 299, 324, 412
TermiNet 300, 441
terminal
  clear screen, 66
  dbx input, 99
  default, 412
  display file on, 298, 323
  display name, 505
  initialization, 499
  messages on, controlling, 289
  name, write, 571
  option setting, 440
  plotting, 330
  process display program, 342
  reserve, 247
  setting characteristics, 499
  show name, 559
  type, ul, 506
  typescript, 402
terminal width, override, 342
terminate, login shell, 252
termination, process, 233, 234
test, Concentrix-2800 command, 481
test-and-loop control structure, 556, 558
testing printers, 265
tex, and lpr, 258
Texas Instruments, 441
text
  dump ELF file, 335
  editor, vi, 540
  formatter, troff, 497
  formatting, nroff, 316
text editor
  See also dired
  ed, 127
  edit, 128
  ex, 149
  GNU emacs, 133
  stream (non-interactive), 403
text file, SCCS, 190
tftp, Concentrix-2800 command, 483
thesaurus, diction, 158
thread_id
  kill
    Bourne shell, 233
    C shell, 234
  mon, 297
  ps, 344
ti700, stty, 441
tilde, stty, 443
time
  and execute, 154
  Concentrix-2800 command
    Bourne shell, 487
    C shell, 485
  execution, later, 19
  set/show, 97
  system, has been up, 522
  variable, 409
time and date, hexadecimal, 96
times, Bourne shell special command, 488
timing, of command execution, 154
tip, Concentrix-2800 command, 489
title, batch request, 351

tmac.bib, 384

tn300, stty, 441

token code, yacc, 578

topological sort, 504

tostop, stty, 443

touch, Concentrix-2800 command, 493

tr, Concentrix-2800 command, 494

trailing, blanks, 116

transferring files, remote machines, 483

translation, by character, 494

translit, m4, 269

transliteration, arguments, 269

transpose axes, graph, 202

trap, Bourne shell command, 495

tree, search directory, 167

troff
  and lpr, 258
  and refer, 367
  Concentrix-2800 command, 496
  delimiter error check, 58
  mathematics preprocessor, 144
  remove constructs, 111
  send output to, 285
  table format, 459

true
  Concentrix-2800 command, 498
  return, find, 167

truth value, Bourne shell, 498

tset, Concentrix-2800 command, 499

tsort, and ar, 16

tsort Concentrix-2800 command, 504

tty, Concentrix-2800 command, 505

tty33, stty, 441

tty37, stty, 441

ttyname, write, 571

turbo processor
  ACC, 152
  disable ACC, 152

type, size, eqn, 144

type casts, C, 244

typescript
  append with script, 402
  terminal session, 402

typesetting
  mathematics, 144
  text, 497

U

Using the mouse, GNU emacs, 137

%U%, get, 194

u, Concentrix-2800 command, 523

ul
  and page, 323
  Concentrix-2800 command, 506

umask, Concentrix-2800 command, 507, 508

unalias, Concentrix-2800 command, 509

uname, find, 167

unbuffered output, cat, 48

uncompact Concentrix-2800 command, 510

uncompress, Concentrix-2800 command, 511

uncompress a file, 511, 585

uncompress file, 510

unconditional, branch control structure, 198

undefine, m4, 269

underlining, 69
  colcrt, 69
  convert for terminal, 506
  suppression, 298, 323
  ul, 506

underscore, in variable names, 238

undivert, m4, 269

unedit, SCCS command, 399

unexpand Concentrix-2800 command, 513

unhash, Concentrix-2800 command, 514

unifdef, Concentrix-2800 command, 515

uniq Concentrix-2800 command, 517

units Concentrix-2800 command, 518

/unix, 342

unknown command, nroff/troff, 58

unlimit, Concentrix-2800 command, 519

unlink
  directory, 380
  file, 377

unpairable line, join, 229

unset, Concentrix-2800 command, 520

unsetnv, Concentrix-2800 command, 521

up, and more, 298

uptime
  Concentrix-2800 command, 522
  remote machines, 388
  sort, 389

usage, summarize disk, 125

USER, environment variable, 413

user
  current, 523
  id, whoami, 560
  information, finger entry, 170
  monitoring, 293
  process, create new, 249
  remote, 375
  ruptime, 389
  rwho, 392
  show current, 559
  uname, 167
  variable, 409
  write, 547, 571

user environment, windows, 562

user information, changing it, 327

user interface, Bourne shell, 414

user time, Bourne shell command, 488

userid, substitution of, 445
username, mail concerning batch request, 351
users Concentrix-2800 command, 523
utilization, monitoring, 293
/etc/utmp, 559
uucp
  Concentrix-2800 command, 524
  mail recieved via, 378
  maintain transaction log, 529
UUCP hosts' names, list, 530
uudecode Concentrix-2800 command, 527
uuendecode Concentrix-2800 command, 528
uulog Concentrix-2800 command, 529
uuname, Concentrix-2800 command, 530
uusend Concentrix-2800 command, 532
uux
  Concentrix-2800 command, 533
  maintain transaction log, 529

V

%V, addbib, 3
vacation, Concentrix-2800 command, 536
vacations, handling mail during, 536
val, Concentrix-2800 command, 538
validation, of SCCS file, 538
variable
  environment, 337
    PRINTER, 258
    remove, 521
    setting, 412
  external, and ld, 238
  list of, 408
  local, remove, 520
  members, shift, 422
  numeric, local, 23
  predefined, 409
  substitution, 87
Varian, 259
verbose
  csh, 83
  tip, 492
  variable, 409
Versatec, 384
  plotting on, 331
vi
  Concentrix-2800 command, 540
  invoking from edit, 128
  invoking from ex, 149
  startup commands, 412
  using with error program, 145
video, display program, 298, 323
virtual memory, statistics, 542
vmstat, Concentrix-2800 command, 542
vt05, stty, 441
VT05 terminal, 441

W

%W%
  get, 195
  SCCS command, 400
w, Concentrix-2800 command, 543
wait, Concentrix-2800 command
  Bourne shell, 546
  C shell, 545
wall Concentrix-2800 command, 547
wc, Concentrix-2800 command, 548
weof, mt, 303
werase, stty, 443
what
  Concentrix-2800 command, 549
  SCCS command, 399
whatis, 14
  Concentrix-2800 command, 550
whatisin, Concentrix-2800 command, 551
whereis, Concentrix-2800 command, 553
which, Concentrix-2800 command, 555
while
  exit from (Bourne), 41
  resume execution after, 40
while (Bourne shell), Concentrix-2800 command, 558
while (C shell), Concentrix-2800 command, 556
while command, sh, 415
who, Concentrix-2800 command, 559
whoami Concentrix-2800 command, 560
whois, Concentrix-2800 command, 561
width, graph, 202
window, Concentrix-2800 command, 562
windowing environment, 562
word
  count, 548
  describe, 550
  documentation files asociated with, 285
  stem, spell, 431
working directory, change
  cd, 52, 53
  chidr, 56
write
  and mesg, 289
  Concentrix-2800 command, 571
  file, 456
  to all users, 547
  to another user, 571
writing, style analysis, 444
writing options to header, execute -w, 154
/usr/adm/wtmp, 559

X

X window system, GNU emacs, 135

%X
addbib, 3
roffbib, 384

xget Concentrix-2800 command, 572

xproc, command, 573

xsend Concentrix-2800 command, 575

xstr Concentrix-2800 command, 576

Y

%Y, addbib, 3

%Y%
get, 194
SCCS command, 400

y, output repeatedly, 579

yacc, 244
Concentrix-2800 command, 578

yellow pages
key map, 582
map master, 584
server, 584

yes Concentrix-2800 command, 579

yet another compiler-compiler, 578

ypcat, Concentrix-2800 command, 580

ypchfn
Concentrix-2800 command, 583
change full name, 583

ypchsh
Concentrix-2800 command, 583
change login shell, 583

ypmatch, Concentrix-2800 command, 582

yppasswd, Concentrix-2800 command, 583

ypwhich, Concentrix-2800 command, 584

Z

%Z, addbib, 3

%Z%
get, 195
SCCS command, 400

zcat, Concentrix-2800 command, 585

READER'S COMMENTS

Please use this form to note suggestions for improvement, or any errors you find in this manual. Alliant Technical Publications Department will try to correct any errors in the next revision of the manual.

Comments on ***Concentrix-2800 Commands and Applications*** **(D002.03)**

Page	Description

Name _____ **Date** ___ _____

Company _____

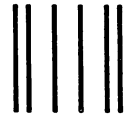
Street _____

City _____ **State** _____ **Zip** _____

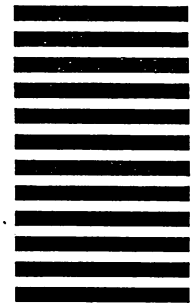
Telephone _____ **Email** _____

Outside USA, return free of charge to your local Alliant Customer Service office. You can also use email; send comments to docmgr@alliant.com.

Fold Here and Tape ----------



No postage
Necessasry
if mailed in the
United States



BUSINESS REPLY MAIL

FIRST CLASS PERMIT # 118 LITTLETON MA 01460

POSTAGE WILL BE PAID BY ADDRESSEE



ALLIANT

ComputerSystemsCorporation

Technical Publications Dept.
One Monarch Drive
Littleton, Massachusetts 01460

-- Fold Here - Do not tear ---------- cut along dotted line this side >>>