Berkeley Unix summary

Applies to 4.2bsd, using the C Shell

All documentation for the system is online. By using the "apropos" and "man" commands, you can usually find anything. Look at the general command format and conventions of this document near the end.

Control Characters

You can change the control characters with "stty", but these are the defaults:

Editing

<rubout>	delete previous character from screen
^W	delete previous word from screen
^U	delete whole current line from screen

Job Control

^C	kill the current process
^Z	stop the current process, but leave it around

End of File

^D	end of file - if you are reading a file from the terminal, this ends it. If you are at command level, it will log you out (but see section on controlling your job

File and Directory Commands

cat files	conCATenate: copies files to std. output as one file
cd directory	Change working Directory
chgrp group files	CHange GRouP id for files. Takes group name or number
chmod mode files	CHange MODe for files. [This is file protection.] E.g. "chmod go+rx foo" allows Others to Read. first letter from "ugo" [user, group, others] symbol is + to add, - to remove, = to set second letter from "rwxst" [read, write, execute, setuid, sticky]
cp infile outfile	CoPy file
cp files directory	CoPy several files into a different directory
-i	interactively confirm if will overwrite an old file
-r	recursively copy all files in directories
diff file1 file2	shows DIFFerences between files
-b	ignore blank spaces
-c	show more of the context
df	shows amount of Disk space Free
du [directory]	shows Disk Usage of directory [default: current dir]
emacs file	edit file with EMACS. (Try "teach-emacs" to learn about EMACS.)
fpr	Fortran PRint (interp carriage ctl) std in --> std out
grep pattern files	search all files for a string (actually a general pattern) Many options. fgrep and egrep are minor variants.

head [files]	show HEAD (first few lines) of files [default std in]
ln oldfile newname	LiNk files: create an alternate name for a file
ln -s filespec name	symbolic LiNk: "name" will refer to filespec
lpq	show Line Printer Queue
lpr files	print files on Line PRinter [has many options]
lprm	ReMove current job from Line Printer [if yours]
lprm -	ReMove all your jobs from Line Printer
lprm jobnumber	ReMove specified job from Line Printer
ls [files]	LiSt files [default: all files in directory]
-a	all files [normally files starting with . not shown]
-l	long form - give lots of information
-t	time-sorted - most recent first
-R	Recursively look into directories
more [files]	print files, stop when screen full [default: std in]
mv oldfile newname	MoVe or rename file
mv files directory	MoVe files into a directory
-i	interactively confirm if new name exists
-	kludge to handle files that begin with "-"
popd	POP Directory name from directory name stack
pushd directory	PUSH current Directory onto dir name stack and then change working directory to specified one
pwd	Print Working Directory name
quota	show your disk QUOTA
rm files	ReMove files (actually only 1 link to them)
-i	interactively confirms each one
-r	recursively remove contents of directories
-	kludge to handle files that begin with "-"
rmdir directory	ReMove DIRectory
tail [file]	show TAIL (last few lines) of a file [default: std in]
touch files	update last write date of files to current date-time

Compiling Programs

cc files	Compile C files. Normally loads as a.out
-c	don’t load, leave relocatables as foo.o
-gx	produce info for debugger [always use this]
-O	optimize
-o output	name the output file this
f77 files	Compile Fortran files. Normally loads as a.out. Can handle several languages: .f Fortran .F Fortran but through C preprocessor first .r Ratfor .e EFL .c C

	.s assembly
-c	don’t load, leave relocatables as foo.o
-C	put in code to check array bounds
-g	produce info for debugger [always use this]
-O	optimize
-o output	name the output file this
pascal files	Compile PASCAL files. Normally loads as a.out
-c	don’t load, leave relocatables as foo.o
-g	produce info for debugger [always use this]
-O	optimize
-o output	name the output file this

Finding Out Information

alias	list all defined commands
apropos word	find all commands and subroutines whose descriptions include this word
date	print current DATE and time
df	shows amount of Disk space Free
du [directory]	shows Disk Usage of directory [default: current dir]
finger	who is logged in, how long idle, office addr and phone
finger user	shows personal information [use chfn to set yours up]
groups [user]	what GROUPS does user [default: you] belong to?
history [number]	print most recent number command you did. If you want to do this, you must put "set history = n" in your .cshrc file, where N is the number of commands you want remembered
mail	send and read mail. Too complex to describe here
man [section] name	look up command, routine, etc. in manual. If section specified, look only in that section.
printenv	PRINT your ENVironment variables (terminal type, etc.)
ps [options]	show processes. default is only yours
a	all processes controlled by a terminal
g	show group leaders (top level processes)
tx	processes on tty x, e.g. ti03 for ttyi03
u	user oriented output
x	even processes with no terminal
pwd	Print Working Directory name
quota	show your disk QUOTA
stty	show current terminal settings
time command	print execution TIME, etc., of a command
uptime	amt of TIME system has been UP, load avg, etc.
users	compact list of all USERS logged on
vnews	read system news/announcements

w	Who is on system, when logged in, load avg., etc.
whatis name	gives you title line from thing's manual entry
whereis name	finds location of system files
who [name]	who is on the system, when logged in, terminal

Setting Things Up

.cshrc	a file in your home directory. Is obeyed by the shell whenever one starts.
.login	a file in your home directory. Is obeyed by the shell when you log in (after .cshrc if any).
.logout	a file in your home directory. Is obeyed by the shell when you logout.
alias name command alias name alias	define a new command undefine a command list all defined commands
clear	CLEAR terminal screen
ignoreeof	don't logout when you type ^D. You must use the logout command. We recommend putting this in .login.
passwd	change your PASSWorD. Asks for old and twice for new
reset	RESET terminal [in case EMACS crashes. Normally you must do "<linefeed>reset<linefeed>", and it may not echo.
script file -a	put SCRIPT of terminal session in a file append to existing file
setenv variable value	set a variable in your "environment", e.g term type Use "printenv" to see what sorts of things there are
set term = v55	tell the system you are using a v55 terminal. You must put spaces around the =. Required on dialup or network lines. (The system knows about hardwired lines.)
stty options stty	set up your terminal. Too many options to list here. show current options
su username	become another user [use ^D to exit]
tset	put in .login file to set up terminal depending upon speed, tty number, etc. Too complex to describe here.

Communicating with Others

ftp	File Transfer Program - copy files over network
postnews	POST an article in the NEWS system (will prompt you for the information needed)
talk person [ttyname]	establish two-window TALK link. other person must also type the command first. End with ^C.
telnet host	login on remote host. ^[q to quit.
vnews	read system news/announcements

write user [ttyname]	WRITEs line from your terminal to his. end with ^D.
! line	! at beginning of line causes it to be executed

Controlling Your Jobs

^C	kill current job
^D	unless you have does "ignoreeof", ^D will log you out (or exit from recursive shell)
^Z	stop current job, but leave it around. (use "fg" or "bg" to continue it, "jobs" to see it)
clear	CLEAR terminal screen
logout	log off the system
reset	RESET terminal [in case EMACS crashes. Normally you must do "<linefeed>reset<linefeed>", and it may not echo.
bg [%job]	continue a job in background. [Default: current job]
csh args	explicit call to the C Shell. Too complex for here.
history [number]	print most recent number command you did. If you want to do this, you must put "set history = n" in your .cshrc file, where N is the number of commands you want remembered
kill 123	KILL process 123 [system-wide numbering, use "ps"]
kill %1	KILL job 1 [your process 1, use "jobs"]
-9	don't let it trap the interrupt
fg [%job]	continue a job in foreground. [Default: current job]
stop [%job]	stop a job [default current] running in background
%job [&]	continue a job in foreground [in background if &]

Conventions in this Document

files	a list of file names, separated by spaces, with possible * and ? as wildcards
%job	% followed by a job number or the name of the program running in the job. Use the "jobs" command to list your jobs. The one with + by it is the "current job". NB: The term "job" is used in the C Shell to refer to a subprocess running under the control of your shell. This is a bit confusing, since the word "job" normally refers to everything a given user is doing. (Indeed the term is used elsewhere in Unix with this meaning.)
-x	options are listed below the command they apply to. Several options can be combined with a single -. E.g "ls -lt". The options are always given right after the command name, before any other arguments.

General Command Structure

command [-options] arguments [<inputfile] [>outputfile] [>&erroroutfile] [&]

command	is the name of a program. The "PATH" variable shown by "printenv" shows you where it looks if you don't specify a directory name. Normally your own directory is in the path, so you can run a program you have compiled just by typing its name.
options	are normally single letters, with a "-" before them

arguments — are normally file names, separated by spaces, but can be other things

< > >& — cause standard input, standard output, and error output to be redirected to files. Not all commands use standard input or output. Typically these represent what you would expect to appear on the terminal. Some commands let you specify a file name, and read from that file, but if you don't specify one, they read from standard input.

& — at the end of a command causes it to be done in the background. You can type other commands while it is being done.

File and Directory Names

The syntax for files, directories, and devices is the same:

    a/b/c/d...

where a, b, c, d, are successive levels in the directory hierarchy, starting with the current directory. So

    prog.c

is a file in the current directory

    bigsystem/prog.c

is in a subdirectory contained within the current directory

    bigsystem/source/prog.c

is in a nested subdirectory, etc.

Dots are just another character, but the tendency is to have a single dot in a file name, and have what is after the dot indicate what the file is, e.g. the language it is in. So ".c" is usually a C program, ".p" Pascal, ".o" a relocatable object file, etc.

If you start the name with a /, then you are giving a directory path starting at the top-level directory (called the "root"). For efficiency reasons, user directories are not directly in the root, but are subdirectories of /u1 or /u2. So one of my files might be

    /u1/hedrick/prog.c

If my current directory is "/u1/hedrick", then I can refer to that file as "prog.c"

Devices are simply special files, normally put in the directory /dev. E.g. your terminal can be referred to as "/dev/tty". If you want to use someone else's terminal, you have to specify the number, e.g. "/dev/ttyi06". The tape drive is "/dev/mt0". Some programs also want access to "raw devices", e.g. the raw tape drive is "/dev/rmt0". This bypasses some levels of the device driver, and is supposedly "more efficient". Don't use a name beginning with "/dev/r" unless the program specifically asks for a raw device. Tape drives are even more complex, as the number encodes (1) which tape drive it is, (2) what speed you want, and (3) whether to rewind it before use.

There are some special characters that you can use in file names:

. — the current directory, e.g. "cp /u1/hedrick/prog.c ." which copies the file to the current directory

.. — the directory above the current one

~hedrick — hedrick's home directory. Since you don't know whether a user's home directory is on /u1 or /u2, it is safest to use this syntax to refer to another user's files.

History Substitutions

These are available if you have turned on the history mechanism, by putting "set history = n", where N is a number, into your .cshrc file. You can repeat a previous command by using

!n — command number [use "history" command to see them]

!-n — n commands ago

!!	previous command
!xxx	command beginning with xxx
!s/old/new/	reexecute previous replacing old with new
! | more	reexecute, piping output through "more"

Actually you can do far more general things, but see the C Shell manual for that.