UNIX Short Reference[1]

C. J. C. Schauble

Beginning UNIX users are often overwhelmed by the number of commands they must learn quickly to perform simple tasks. To assist such users, this document contains a sampling of commonly-used UNIX commands.

Informative Commands: These commands provide information about your login, environment, terminal, machine, and system. They also allow you to make some changes to these states.

`date` – to display the current date and time

`kill` – to kill the process with a given `pid` as argument

`man` – to get information on a UNIX command

- `man f77` – shows the pages of the UNIX manual referring to `f77` on the screen.

`nslookup` – to find the address of a given machine

- `nslookup yourmach` – returns the name and address of the machine `yourmach`, along with the name and address of its server.

`passwd` – to change your current password

`printenv` – to show the current environment setting

`ps` – to list your current processes with their `pid`

`setenv` – to change an environment setting

- `setenv DISPLAY yourmach:0` – tells the Xserver that the Xterminal named `yourmach` is where any windows created are to be displayed.
- `setenv PRINTER xxx` – makes `xxx` be the default printer for any `lpr` commands.

`time` – to time the execution of a given command

- `time anyunixcommand` – executes `anyunixcommand` and returns the user, system, and total time taken for the execution

`who` – to list users currently logged in to given machine

`whoami` – to display login of user currently logged onto given terminal

[1]This work has been supported by the National Science Foundation under an Educational Infrastructure grant, CDA-9017953. It has been produced by the HPSC Group, Department of Computer Science, University of Colorado, Boulder, CO 80309. Please direct comments or queries to Elizabeth Jessup at this address or e-mail `jessup@cs.colorado.edu`.

Copyright ©1995 by the HPSC Group of the University of Colorado

File Manipulation: Remember that UNIX uses a tree structure for storing files. The main root of this tree (for the whole machine or system) is named `/`. Your main (or *home*) directory can be addressed as `~` or `~yourloginname`. The home directory of another user with login `guy` would be `~guy`. The current directory (the one you are working in at the current time) is referred to by a single dot (`.`). The parent directory (the next one up the tree) is represented by two dots (`..`).

All file names in the following examples use capital letters, e.g., `ABC`. Lowercase letters and digits could have been used as well; for example, `9dec91` and `afile` are legal filenames. Remember that UNIX distinguishes between upper and lower case; `XYZ` and `xyz` refer to different files.

`cd` – to change directory

- `cd ABC` – moves to a directory named `ABC` located in your current directory.
- `cd ..` – moves to the parent directory of your current directory.
- `cd` – moves to your home directory.
- `cd ../ADIR` – moves to a directory named `ADIR` located in the parent directory of your current directory.

`cp` – to copy one file to another

- `cp ABC DEF` – copies file `ABC` to (or on top of) a file named `DEF`.
- `cp -i ABC ADIR/DEF` – copies file `ABC` to (or on top of) a file named `DEF` in the directory `ADIR`. Requests approval for overwriting the file if the file `ADIR/DEF` already exists. (`-i` means `i`nteractive.)
- `cp -r ADIR BDIR` – copies the entire contents of the directory `ADIR` to a new (or on top of the old) directory `BDIR`. (`-r` means `r`ecursive.)

`ls` – to display file information

- `ls` – lists all files in your current directory.
- `ls -a` – lists all files in your current directory, including any *dot* (`.`) files (e.g., `.login`).
- `ls -F` – lists files in your current directory, putting a slash (`/`) after those that are directories and an asterisk (`*`) after those that are executables.
- `ls -l` – lists all files in your current directory, showing protection codes, date of creation, and size.

`mkdir` – to make a new directory within current directory

- `mkdir BDIR` – creates a new subdirectory named `BDIR` within the current working directory.

`mv` – to rename (or move) a file

- `mv -i ABC DEF` – renames `ABC` to `DEF`; can also be thought of as moving the file `ABC` on top of file `DEF`, asking permission if the file `DEF` already exists. (`-i` for `i`nteractive.)

`pwd` – to display full pathname of current working directory

`rm` – to remove (delete) a file

- `rm ABC DEF` – deletes both `ABC` and `DEF`.
- `rm -i ABC DEF` – first asks you if you really want to delete these files; then deletes the ones for which you respond yes (`y`). (`-i` for `i`nteractive.)

`rmdir` – to remove (delete) a directory

- `rmdir MNO` – deletes the directory named `MNO`.

Language Commands: These commands help you to compile and debug programs.

`cc` – to compile a C program

- `cc -O acprog.c -o acprog -lm` – compiles with optimization (`-O`) the C program named `acprog.c` into the executable file named `acprog`, allowing the compilation to access the math library (`-lm`).

`dbx` – to debug a program

- `dbx aprog` – runs the executable program named `aprog` that was compiled with a `-g` option in a debugging environment.

`lint` – to check the syntax of a C program

`f77` – to compile a Fortran program

- `f77 -c fprog.f ftn1.f ftn2.f` – compiles, without generating an executable file (`-c`), the Fortran program named `fprog.f` with the additional Fortran modules, `ftn1.f` and `ftn2.f`.
- `f77 -g -o debug anfprog.f` – compiles the Fortran program called `anfprog.f` with a symbol table (`-g`) so that the executable file named `debug` can be used with the `dbx` command.

Displaying Files: These commands allow you to see the contents of a file.

`cat` – to display a text file or concatenate files

- `cat file1` – displays `file1` on screen.
- `cat file1 file2` – displays `file1` followed by `file2` on screen.
- `cat file1 file2 > file3` – creates `file3` containing `file1` followed by `file2`.

`diff` – to show the differences between two files

- `diff ABC DEF` – displays any lines in `ABC` or `DEF` that differ from each other.

`lpr` – to print a file

- `lpr -Pxxx ABC DEF` – prints out files `ABC` and `DEF` on printer `xxx`.

`more` – to display a file, screen by screen

- `more ABC DEF` – displays the two files `ABC` and `DEF` sequentially on the screen. Hitting the space bar gets next screen; the return key gets the next line.

`pr` – to paginate a file before printing it

- `pr ABC DEF` – breaks both `ABC` and `DEF` into pages, puts a heading on the top with the name of the file, the date and time, and a page number. The two files are numbered independently. The result goes to the screen.
- `pr ABC | lpr -Pxxx` – paginates the file `ABC` and sends the resultant file to be printed on `xxx`. This is an example of a UNIX command that uses a *pipe* ("|"); that is, the standard output of the first part of the command (before the pipe "|") is `pipe`d to (is treated as the standard input for) the second part.