Colorado State University
Computer Science Department

Basic Unix Commands

What is Unix?

Unix is the name of an operating system used by most supercomputers and workstations today. This system was originally developed by AT&T and further expanded at Berkeley. Several versions exist today with the same basic core of commands. For instance, Solaris is a version of Unix developed by Sun Microsystems (the company that also developed Java), HP/UX is the name of the Unix version for Hewlett-Packard workstations, ULTRIX is the version used by Digital Equipment, Inc, and UNICOS runs on the Cray supercomputers.

Beginning Unix users are often overwhelmed by the number of commands they must learn quickly in order to perform simple tasks. To assist such users, this Web page contains a sampling of commonly-used Unix commands to allow you to perform some basic and necessary operations on most any Unix system.

If you require more information about any of these (or other) Unix commands, then type `man` *anyunixcommand* where *anyunixcommand* is the command in question. The system should display the on-line documentation for the *anyunixcommand*. Don't become concerned if some of these commands do not make sense to you; they should become more meaningful as you become more experienced with the Unix environment.

NOTE: Unix is a **case-sensitive** environment. Be sure not to use a capital letter in place of a lowercase letter; the results will not be what you expect.

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. These are listed in alphabetic order, not in order of importance.

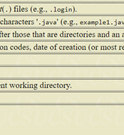
`date`	*to display the current **date** and time*	
`kill`	*to **kill** (or destroy) the process with a given `pid` (process identification number) as argument*	
`logout`	*to **log out** from the Unix system*	
`man`	*to get information on a Unix command; to look up the page in the online **man**ual for that command*	
	`man CC`	shows the pages of the Unix manual referring to the C++ compiler (`CC`) 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 **env**ironment setting*	
`ps`	*to list your current **p**rocesses by their `pid` (process identification number)*	
`setenv`	*to change an **env**ironment **set**ting*	
	`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` or `enscript` commands.
`source`	*to reexecute a **source** shell script file*	
	`source .login`	re-executes your `.login` file (normally executed when you log in); useful after making changes to the .login file (removing the need to exit and re-login);
`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 the users currently logged in to given machine; to find out **who** is logged in*	
`whoami`	*to display login of user currently logged onto given terminal; to answer the question: "**Who am I**?"*	

File Manipulation:

A basic understanding of the Unix file system is required. Unix allows files in your *login* directory (the directory you exist in when you successfully login to a Unix system); Unix also permits the creation of subdirectories to contain files.

The Unix operating system 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 (**..**).

Now suppose your loginname is `stu` and consider the following set of files:



In the above structure, the subdirectories existing under the login directory (`~stu`) are shown; these subdirectories are named `A`, `B`, `C`, `V`, `W`, `X`, `Y`, and `Z`. Note that `A`, `B`, and `C` are subdirectories of `~stu` . Also note that

- `V` and `W` are subdirectories of `A`;
- `B` has no subdirectories;
- subdirectory `C` has subdirectories `X`, `Y`, and `Z`.

Any number of files (or none at all) can exist in any subdirectory.

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 **d**irectory*	
	`cd ABC`	moves to a subdirectory named `ABC` located below 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 **c**o**p**y one file to another; copies the contents of the first file (or directory) to the second file (or directory), without changing the contents of the first file (or directory)*	
	`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 interactive.)
	`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 recursive.)
`expand`	*to **expand** the tab characters found in one file into spaces in a new file*	
	`expand AFILE BFILE`	copies the contents of `AFILE` into a new file called `BFILE`, changing all tab characters to spaces.
`ls`	*to display file information (to list **s**ub**d**irectory); lists only the file names of the files in the current subdirectory*	
	`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 *.java`	lists all files in your current directory that end with the characters '`.java`' (e.g., `example1.java`).
	`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 (or most recent modification), and size.
`mkdir`	*to **m**a**k**e a new sub**dir**ectory within (or below) the current directory*	
	`mkdir BDIR`	creates a new subdirectory named `BDIR` within the current working directory.
`mv`	*to rename (or **mov**e) a file; renames the first file (or directory) named to the name of the second; the first file (or directory) no longer exists*	
	`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 interactive.)
`pwd`	*to display full **p**ath**n**ame of current working sub**d**irectory; to provide the name of the subdirectory you are currently working in*	
`rm`	*to **r**e**m**ove (delete) a file; once deleted, this file is **unretrievable** (unlike the Trash Bin facility on PCs and Macs)*	
	`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 interactive.)
`rmdir`	*to **r**e**m**ove (delete) a sub**dir**ectory; the subdirectory cannot contain any files; you need to `rm` (delete or purge) them first*	
	`rmdir MNO`	deletes the empty subdirectory named `MNO`.

Java Programming Language Commands (for JDK):

These commands help you to compile and execute programs and applets in Java using the Java Development Kit (JDK).

`javac`	*to compile a Java program*	
	`javac -g acprog.java`	compiles with debugger information (`-g`) the Java program named `acprog.java` into the Unicode (bytecode) file named `acprog.class`.
`java`	*to interpret and execute a compiled Java program*	
	`javac aprog`	interprets and runs the Java bytecode file named `aprog.class`. Notice that the '`.class`' extension is assumed by the `javac` command.
`jdb`	*to debug a Java program*	
	`dbx aprog`	runs the Java bytecode file named `aprog.class` that was compiled with a `-g` option in a debugging environment. Again the '`.class`' extension is assumed by the `jdb` command.
`appletviewer`	*to run a Java applet*	
	`appletviewer anapplet.html`	executes an applet that is called from an URL (or HTML file) named `anapplet.html`.

Other Language Commands:

These commands help you to compile and debug programs in other programming languages.

`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`).
`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 and Printing Files:

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

`cat`	*to display a text file or to con**cat**enate files*	
	`cat file1`	displays contents of `file1` on the screen (or window) without any screen breaks.
	`cat file1 file2`	displays contents of `file1` followed by `file2` on the screen (or window) without any screen breaks.
	`cat file1 file2 > file3`	creates `file3` containing `file1` followed by `file2`.
`diff`	*to show the **diff**erences between two files*	
	`diff ABC DEF`	displays any lines in `ABC` or `DEF` that differ from each other.
`enscript`	*to print a file with filename, date, and page number*	
	`enscript -Pxxx -2rG ABC`	prints out the contents of file `ABC` on the printer named `xxx` with two columns per page (`-2`), rotated 90 degrees (`-r`) so that it appears in a *landscape* format, with a *gaudy* heading (`-G`) as a shaded bar across the top that provides the filename (`ABC`), the creation date of that file, and the page number.
`lpr`	*to **p**rint a file*	
	`lpr -Pxxx ABC DEF`	prints out the contents of the file `ABC` followed by the contents of the file `DEF` on printer `xxx`.
`more`	*to display a file, screen by screen; to list the contents of a file to the terminal screen (or window)*	
	`more ABC DEF`	displays the two files `ABC` and `DEF` sequentially on the screen. Hitting the space bar moves down one screen; the return key moves down one line.
`pr`	*to paginate a file before printing it (to **pr**etty it)*	
	`pr ABC DEF`	breaks the contents of the files `ABC` and `DEF` into pages, puts a heading on the top of each file 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 piped to (is treated as the standard input for) the second part.
`spell`	*to perform a spelling check on a file; to list words found in the file that are not in the Unix spelling dictionary; **Note:** often lists words that are hyphenated (split across two lines)*	

For further information, see the following Unix tutorials:

- Introduction to Unix from Maui High Performance Computing Center
- Unix tutorial from England
- Unix tutorial from UC Davis
- Unix tutorial from BYU
- Unix tutorial from Cal Tech
- Unix tutorial from Idaho State University