Unix

Although there is a well-defined technical and commercial standard for what constitutes "Unix," in common usage, Unix refers to a set of operating systems, from private vendors and in various open-licensed versions, that act similarly from the view of users and administrators. Within any Unix version, there are several different "shells" which affect how commands are interpreted. Your default for this course is that you are using Solaris (developed by Sun Microsystems primarily for use on hardware sold by Sun). Most of the basic commands here will work the same in other Unix variants and shells, including Linux and in the command-line environment of Mac OS X.

All Unix commands and file references are case sensitive: "This" is a different filename than "this" because of the capitalization difference. All Unix commands are lowercase and from two to nine characters long. Many commands have options that are invoked by a hyphen followed by one or more letters. Multiple options can often be requested by adding multiple letters to a single hyphen. For example, ls -al combines the -a and -l options.

Users and Resources

who Find out who else is logged on.

finger Find out who else is logged on—different format.

finger *name* Find information about anyone with *name* in their user name or real name.

finger -m *username* Find information about person whose user name is *username*.

chdgrp Find out what project numbers you are a member of.

newgrp *project* Change your project number to *project* for the current login session.

quota -v Find out how much disk space you have in use and available.

uptime Find out what time it is, how long since the computer last rebooted, how many users are logged on, and how many processes are waiting for a processor.

A standard Unix system provides commands username, passwd, chsh, and additional options on chdgrp to change usernames, passwords, default groups, and shell environments. UD has modified it's central Sun system so that all such requests, as well as requests for resources such as additional disk space, can be made using web forms at http://www.udel.edu/network. On a Macintosh, use the Accounts preference pane for these activities.

File Handling

ls List the files on the current directory.

ls -l List files in long form, showing time and date they last changed, their size, and access permissions.

ls -a List all files, including "initialization" files whose names begin with a period.

cp *filename1 filename2* Copies existing *filename1* to new *filename2*.

cp *filename1 directory* Copies *filename1* into *directory*, retaining the same filename.

mv *filename1 filename2* Rename existing *filename1* to be called *filename2*.

mv *filename directory* Move *filename* into a subdirectory *directory*.

rm *filename* Permanently erase (remove) a file.

rm -r *directory* "Recursively" erase a directory and everything it contains, including subdirectories. (Use with extreme caution.)

Wildcards: * is a "wildcard" character that can refer to any character string and ? is a wildcard character that can refer to any single character. E.g., mv *.f95 code would move every Fortran 95 program file on the current directory into a subdirectory called code. These wildcard characters are a subset of a larger set of rules for building character-string patterns called "regular expressions."

Filenames: in our version of Unix, they may be up to 255 characters, and they may include any character except the regular slash /. (Avoid using backslashes, blank spaces, or nonprinting characters in filenames—they are allowed but will cause difficulties.)

Directories

Unix allows you to organize files into a directory tree, exactly analogous to the use of "folders" in Windows or Mac OS filesystems.

cd "Change Directory" to your home directory.

cd */absolute/path/name* Change to the directory indicated – path begins with /.

cd *subdirectory* Move down the directory tree into subdirectory – path does not begin with /.

cd .. Move up one step in the subdirectory chain.

pwd "Print Working Directory," show the complete path to the current working directory.

mkdir *newdirectory* "Make Directory," makes a new subdirectory below the current directory.

rmdir *emptydirectory* "Remove Directory."

A pathname beginning with / is an **absolute path** from the top of the system tree. A pathname not beginning with / is a **relative path** down from the current working directory. Directory shortcuts include: ~ as a replacement for your home directory, ~*username* as a shorthand for *username*'s home directory, .. (two periods) for the subdirectory one level up from the current directory, and . (one period) for the current directory.

Miscellaneous useful commands

cat *filename* Type a file called *filename* to the screen.

more *filename* Type a file called *filename* to the screen one screen at a time. Actually, any command can be "piped" into more so that the output appears one screen at a time. For example, if a directory has very many files, you can type ls | more and get the file listing one screen at a time.

date Find out time and date.

cal See a calendar for the current month. Add a year number to get a whole year.

A command repeatedly used with many options can be replaced by an **alias**. You have file called .alias on your account. Add lines to it of the following form if you wish to customize your Unix account.

alias *newalias command* Defines *newalias* as a short cut for the previously existing Unix command. A useful example:

alias clean 'rm *~ *#' creates an alias clean, so that subsequently the command clean will remove every Emacs backup file from a directory.

History

Unix keeps track of recent commands you have issued. These commands can be recalled and rerun, or just looked at to help you figure out what you've done.

history Prints the list of your recent commands, with a sequence.

!*number* Rerun command number *number* from the history list. For example, if you use history to get your command list and find that the 38th command you issued in this session was rm *.f95~ to clear out all your XEmacs backup files, then typing just !38 is equivalent to retyping that entire command.

!*letters* Rerun the command from the history list that starts with *letters*. A very common and useful example for this course is for recompiling a program you've been working on. Suppose you compile a program once using f95 exercise2.f95, and it has errors you need to fix. Fix the errors as best you can in your editor, then if you enter the command !f, that will tell Unix "rerun the last command that started with f" and it will recompile the program.

You can also use arrow keys to browse through and rerun recent commands in either `tcsh` or `bash` shells. Another useful way to shorten commands is **tab completion**, which is easier to demonstrate than to explain.

Process Control

When you type a command at the Unix prompt, press return, and wait until the prompt comes back indicating the previous command is done, then you have run a **foreground process**. You can only run one foreground process at a time from one window, but Unix allows you to run more than one process at once, some of which are in the **background**.

To start a long program running (one that may take several minutes to complete, for example) put it in the background by adding a **&** to the command.

./a.out & Runs the compiled Fortran program a.out as a background process, which means that the prompt comes back as soon as you start the job and you can do other things while a.out runs.

xemacs *something*.f95 & Starts an editing session on *something*.f95 as a background job. Since XEmacs will create a new window separate from the terminal window in which it is invoked anyway, this is the most useful way to edit things on an X-window session. When debugging a program, you can save a file during the editing session and then compile it from the Unix window without exiting XEmacs each time.

control-Z, bg, fg If you start a process in the foreground and it appears to take longer than you want, pressing control-Z will suspend the process (stop it temporarily). To restart the process in the background, type bg. You can subsequently bring it back to the foreground with fg. In some X window applications, including XEmacs, control-Z "iconizes" the application; you can restore it by double-clicking the icon.

control-C You can terminate any foreground process with control-C. This can be very useful if you accidentally create an infinite loop in a Fortran program.

ps This produces a list of processes that are currently running under your control. The important columns in the listing you get are the process number (PID), how much processor time it has used (TIME) and the COMMAND that started the process.

kill -9 *pid* You can terminate any background process with kill and the process number. You can find out the process number *pid* of your current jobs with ps. The -9 option is needed, kill alone does not usually make things dead enough.

Help

The built help available for Unix commands is generally poor for new users, because it consists of complete technical information explaining every option to the advanced user.

man *command* Displays the complete Unix manual pages for *command*. Often, the information is so complete as to be confusing.

apropos *keyword* Lists any commands that contain *keyword* in their short summary; used to find a command when you know a task.

More useful than the man pages typically is the Web. Googling a Unix command name will almost always produce a Wikipedia entry in the first few links.

Environment Variables and Shells

Environment variables are an advanced topic that should be avoided in an introductory course. A set of environment variables has been created for you in your setup procedures that enables Unix to find the software and libraries you need to get through this course. How environment variables are set, used, and handled, varies with the shell. Only if something goes wrong, you may be asked to issue commands like this:

env List all the environment variables and their values.

echo $VAR_NAME List the value of the environment variable called VAR_NAME, as an example. The dollar sign is necessary when a variable's value is being requested.

setenv VAR_NAME *new variable value*
Sets a new value for an environment variable in C-Shell.

unsetenv VAR_NAME This undefines and unsets an environment variable in C-Shell.

export VAR_NAME=*new variable value*
Sets a new value for an environment variable in Bash.

unset VAR_NAME This undefines and unsets an environment variable in Bash.