Unix on a Mac

Unix is an operating system that was first developed about 40 years ago for use on small computer systems. Over the decades, it has become a fundamental underpinning of systems from the iMac computers in our lab to the world's largest supercomputers. It is very flexible and capable of being very robust and secure, but it may feel a little old-fashioned at first.

Unix encompasses several "layers" of software, only one of which will be used by most users. The **shell** refers to a an interactive system in which a user types text commands, presses the $\boxed{\text{return}}$ key, and the computer responds, usually with more text. This is called a **command line interface** (CLI). Every thing you will do with typed commands in the shell has an analog in the **graphical user interface** (GUI, often pronounced "gooey") that uses a mouse or trackpad to drag, click, or otherwise manipulate icons arranged in folders that represent files in directories.

In a CLI, folders are **directories**. They are still arranged in a hierarchical "tree," but moving from one folder to another requires a text command, the cd "change directory" command, followed by a directory name or shortcut. Files are still files, but the only way to move, copy, erase, look at their names, rename them, or start them running as programs, is via text commands.

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.

To start a Unix shell session open /Applications/Utilities/XQuartz.app, which should already appear in your dock as a big X with an oval around it.

Looking at Directories and Files

cd	"Change Directory" to the home directory.
cd /*absolute***/***path***/***name*	Change to the directory indicated.
cd *subdirectory*	Move down the directory tree into subdirectory.
cd ..	Move up one level in the subdirectory tree, to the directory that contains the current working directory.
cd /	Move to the top of the directory tree.
pwd	"Print Working Directory," show the complete path to the current working directory.
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.

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 on machines that have more than one user set up, .. (two periods) for the subdirectory one level up from the current directory, and . (one period) for the current directory.

Changing files and directories

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. Note that Unix ”remove” commands do not move things to a wastebasket or recycle bin for you to change your mind later – they are permanent once issued.
mkdir *newdirectory*	“Make Directory,” makes a new subdirectory below the current directory.
rmdir *emptydirectory*	“Remove Directory.”

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 *.py code` would move every Python program file on the current directory into a subdirectory called `code`.

Filenames may be up to 255 characters long 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 in the command-line interface.)

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.

History

Unix keeps track of recent commands you have issued within a session. 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 and time.
!*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*. E.g., if you have run a `python` program using the `python` command and did not get a good result, fix the errors in Emacs and enter the command `!p`, which will tell Unix to rerun the last command that started with p.
↑ ↓	The up and down arrow keys can be used to browse through the list of your recent commands. Once you pull back a command with the arrows, it can be edited before pressing $\boxed{\texttt{return}}$ to execute the command.
$\boxed{\texttt{tab}}$	**Tab completion** works on filenames referenced in commands. If you type the first letters of a filename and press $\boxed{\texttt{tab}}$, the shell will try to complete the file name based on whatever filenames are actually present in the current directory. This is probably easier to demonstrate than to describe.