Unix on a Mac

Aim: This is an an introduction to UNIX on a Mac for beginners. The aim is to give you the basic commands you might use when interacting with the computer via the terminal window.

How to use this document: Try the steps listed enumerated below to learn to use the basic unix commands on a Mac.

1 Terminology

The shell is a command line interpreter or interface. The default is `tcsh` on a Mac, but others are available (including `bash` or `csh`). The choice of shell affects the commands that can be used, i.e. like writing commands in a different programming language.

The directory structure describes how files are organised, with folders and subfolders, and the **root** at the origin (given by the for-slash, `/`). We will look at navigating and changing the directory structure from the Terminal window, making changes like what would otherwise be possible in the Finder.

Shell script A shell script is a collection of unix commands dumped into a file which can then be run by calling the filename.

2 Quickstart: Common commands

On a Mac, the Terminal window program is found under your `Applications/Utilities`. You can either navigate to your `Applications` folder in the Finder window, or look for it in the Launchpad.

2.1 Finding out about where you are

Open a Terminal window on your Mac. Execute the following commands.

1. `pwd` or "print working directory". Use this command to see where you are, e.g. `/Users/eddifying`. This is the full path location.
2. `ls` or "list". Use this command to see what is in the current directory.
3. `ls D*` to see only those files and directories starting with a `D`, where the asterisk (`*`) is a wildcard.
4. `ls -a` or "list including hidden files". `-a` is an option which tells the list command to show hidden files as well, which are files whose names start with a full stop (`.`).
5. `ls -la` or "long list". We've now stacked a second option the `l` which shows various properties of the files and directories. The left-most character is either a `d` or a hyphen (`-`), where the `d` indicates that it's a directory. (The other characters identify the permissions, then the owner of the file, the group of the owner, the size of the file, date last modified, and the name.) From the list, identify a directory.
6. `cd` or "change directory". Using the directory identified, e.g. `Documents`, move into that directory with `$ cd Documents`.
7. `pwd` to see where you are now. Note that the working directory is now the subdirectory of what you've chosen. The subdirectory you've chosen will now be stacked onto the path you were previously in from (1).
8. `cd ..` or go up one level. To navigate backwards in the directory structure, use the double dots (`..`) with the `cd`.

9. `cd /Users/eddifying/Desktop` which directly navigates to a specific location.

2.2 Changing things

Besides just getting information and navigating through the directory. You can create directories, remove directories, delete files, create files, etc.

1. `mkdir dummy` to create a new directory called "dummy"
2. `cd dummy; touch silly; ls -l` to move into the new directory, create a file called "silly" and list the directory contents to see that `silly` is there. (You probably won't need the command "touch", but this also shows how you can stack commands together using the semicolon (`;`).)
3. Note the **permissions** on `silly`. It should read `-rw-r--r--` which means it's a file (not directory), and the owner has read (`r`) and write (`w`) permissions but not executable (`x`). The group (`staff`) and everyone have read (`r`) but not write or execute.
4. Now navigate to the `dummy` directory in the Finder window. Right-click on `silly` and select `Get info`. Near the bottom of the menu, you'll see `Sharing & Permissions` which tells you the same information. In my case `eddifying` has Read and Write, `staff` (the group my username is a member of) has read only and `everyone` has read only.
5. `pico silly` to edit the file "silly". `pico` is a simple text editor. Type something in, then close the file with a `ctrl-x`. Emacs and vim are also available as text editors on a Mac.
6. `more silly` to see what is in the file "silly". Similar commands include `less` and `tail`. You'll need a longer file to see how they work.
7. `cp silly funny` to make a copy of the file `silly` and call it `funny`.
8. `rm silly` to remove or delta the file "silly." Try a `$ man rm` to see the options you have with remove. A few common options are `-f` or force delete (even if files are locked or read-only) and `-r` recursive delete, which will delete a directory and anything in any subdirectories as well. You want to be very *very* careful before executing a `$ rm -rf` which will recursively force-delete everything in the sub-directories of the current location.
9. `$ cd ..; rmdir dummy` to navigate up one directory and then delete the folder "dummy." Note, when you execute this, you'll get the warning `rmdir: dummy: Directory not empty` if the file `funny` is still there. Try an `$ rm -r dummy` to recursively delete dummy and all the files in it.

2.3 Other useful commands

1. `ftp` to transfer files from one machine to another using file-transfer-protocol.
2. `scp` to secure copy files from one machine to another (with a static ip address).
3. `ssh` to secure-shell connect to another machine.
4. `chmod` to change permissions on a file.
5. `grep` to find strings within files.
6. `locate` to locate files.
7. `mv` to move files.
8. `gunzip` and `gzip`, uncompress and compress.
9. `ctrl-c`, the universal kill command.

10. `sudo` is a preface that can be used before any command to execute it as the superuser. This only works if you are an administrator on your computer, and can be useful because it overrides a bunch of options and lets you write in places that your username doesn't own (like above the `/Users` directory).
11. `cat` dumps the contents to the standard input (screen).
12. `alias` name a command or string of commands something else (like shorthand to do something you do often). `alias 'ls'='ls -latr'` will mean that whenever you just type `ls`, it will then call `ls -latr`. However, this command will only hold in the current terminal window. More permanent changes that you want executed every time you open a terminal window should go in one of the dot-files in your home directory.

3 Making a simple shell-script

Try putting it all together by making a shell script.

1. Make a new dummy directory and open a file inside it using `pico`.
2. Enter a few commands into this file, with each command on a new line or separated by a semicolon (`;`). Avoid the `rm` commands for now. Try a few `ls`'s and perhaps a `touch nutty`.
3. Close (and save) the file with a `ctrl-x`.
4. Try running the file with a `$ ./yourscriptname`. You should get the error
   `-bash: ./yourscriptname: Permission denied`.
5. Add executable permissions to the file using `chmod`, then run it again.
6. Verify that what was output to the screen matches the sequence of commands in the script.

4 Finding more help

- To get a basic list of commands available in the shell, type a `$ help` at the command line.
- If you know the command you want, but want to see how to use it/options available, use the man(ual) pages. Try this with the list command, `$ man ls`. For `ls`, I typically always use the `-l` option, and often an `-latr` or maybe an `-latrh`. Look up the options in the man pages to see what these do.

```
$ man ls
LS(1)                    BSD General Commands Manual                    LS(1)

NAME
     ls -- list directory contents

SYNOPSIS
     ls [-ABCFGHLOPRSTUW@abcdefghiklmnopqrstuwx1] [file ...]

DESCRIPTION
     For each operand that names a file of a type other than directory, ls displays its name as well as
     any requested, associated information.  For each operand that names a file of type directory, ls dis-
     plays the names of files contained within that directory, as well as any requested, associated infor-
     mation.
```

- Online tutorial on unix basics: `http://acad.coloradocollege.edu/dept/pc/SciCompLab/UnixTutorial/`, particularly tutorial 3 on pipes, tutorial 4 on wildcards, and tutorial 8 on setting the search path.
- For more detail on shell-scripting, see `http://www.freeos.com/guides/lsst/`