

Unix tutorial

Thanks to Michael Wood-Vasey (UPitt) and Beth Willman (Haverford) for providing Unix tutorials on which this is based.

Terminal windows

You will use terminal windows to enter and execute commands to the operating system. To open up a terminal window on a Mac machine, click on "Applications" in the menu bar at the top of your screen, and enter the "Accessories" submenu. Select "Terminal" from this submenu. You can use the menubar on the terminal window itself to open more than one terminal window, either by using tabs or by opening separate windows.

Directory Structure

The directory architecture used by Unix is a hierarchical tree structure, much like what you may have used in a Windows or other computing environment. When you first log onto a Unix system, you land in your "home" directory. Instead of clicking on folders, you navigate through the directory structure by "changing directories" using the command `cd` at the prompt in your terminal window. The directory that you are currently sitting in at a prompt is your "working directory".

The top of the tree is the "root" directory, designated with a slash sign (`/`). Directory names build on the root directory, with a slash designating a new branch in the directory structure. For example, the full (absolute) path name for my home directory would be `/home/mgeha/`. Directories can also have relative path names, which are interpreted as starting from the present working directory. If we were in the `/home/` directory, the relative path name to my home directory would be `mgeha`.

Listing the Contents of a Directory

The command `ls` is used to list the contents of a directory. Under Unix, files, directories and even devices are treated as philosophically equal, so a directory could contain any one of these types of items. The `ls` command can be invoked with a variety of options, which modify its action. Under Unix, command options are preceded by a hyphen. A few of the more useful forms of `ls` include:

<code>ls</code>	lists all filenames, including directories
<code>ls -l</code>	generates a full listing, including dates, sizes, etc
<code>ls -p</code>	distinguishes between directories and regular files
<code>ls -t</code>	lists contents by date of their creation, latest first.

Try each of these `ls` commands in order and notice the way they indicate your home directory's contents. You can combine different options when executing a Unix command, for example:

`ls -lt`

Creating Directories

Create a new directory in which you will work by using the command `mkdir`. You can call this directory anything you'd like, but avoid whitespaces and try to name it something that makes sense, like "A255".

```
mkdir A255
```

Now list the contents of your current working directory to verify that your new directory has been created.

Directories can be deleted with the `rmdir` command.

Changing Directories

Learn where you are in the directory structure by typing

```
pwd
```

which is an abbreviation for "print working directory".

The `cd` command moves you around in the directory structure, and takes an argument that is the desired destination directory. The argument can be either a relative or an absolute pathname. Type

```
cd yournewdirectory
```

to enter the directory that you've just created and the system will move your current working directory to `/home/yourname/yournewdirectory`.

Unix provides some useful shortcuts for navigating directories. A single dot, `.`, always refers to the current directory. Double dots, `..` refer to the parent directory. You can therefore move into the directory `/home/mgeha` from `/home/mgeha/A255_Data` by simply typing:

```
cd ..
```

You can always get back to your home directory by typing `cd` with no arguments.

Copying Files

The command `cp` is used to copy files:

```
cp source destination
```

so for example I could make a backup copy of the file `foo.txt` by typing

```
cp foo.txt foo.bak
```

Viewing and editing the contents of a file

The tasks `cat file` and `more file` print the contents of a simple text file to the terminal window. The `cat` task scrolls the entire thing past with no pauses. The `more` task `more` allows you to page through the file, using the spacebar to move one screenful and the Enter key to move down one line at a time.

There are a number of ways to edit text files. I use the program *emacs*. Other options are *vi* or *textedit*. We will discuss the pro/cons of these options in class. You open up a file to edit with *emacs* by typing:

```
emacs myfile
```

You can do this whether or not the file yet exists. Its better if you open a file using emacs by typing:

```
emacs myfile &
```

The “&” will run emacs in the background of your terminal, freeing up your terminal prompt for you to execute commands.

Go to the directory that you created for this class. Use emacs to create a file in which you will keep notes. Write a couple of notes, and save and close the file.

Edit your .bashrc File

I want you to add an alias to the `.bashrc` file in your home directory that will help prevent you from accidentally deleting files that you want. Add the line:

```
alias rm='rm -i'
```

to your `.bashrc` file. Save and quit this file.

Filenames and Wildcards

Files in the current working directory can be referred to simply by their filename. In general, however, the name of a file includes its full pathname, such as

```
/home/mgeha/foo.txt
```

where the last element, “foo.txt”, is the file name.

Unix allows you to select a subset of the objects of interest. For example, to obtain a listing of all files that end in “.doc”, you would type

```
ls *.doc
```

You could also move all files that fit a particular pattern with

```
mv ../*.doc .
```

Redirection

One of the most useful aspects of Unix is the ability to redirect streams of information, input and output to and from files and processes. The symbols `>`, `<` and `>>` are used to accomplish this. For example, to send the listing of all the files in the current directory into a file called `foo`, one would type

```
ls > foo
```

This has the feature that if a file called `foo` existed already, it would be overwritten with the new information. You could instead append the directory listing to a pre-existing file called `foo` by using

```
ls >> foo
```

Online help under Unix

The on-line documentation for Unix can be accessed through “man pages”, short for manual. If you remember the relevant work, you can type

```
man -k keyword
```

Which will produce a listing of all the man pages that pertain to that topic. You can then type

```
man foo
```

to get specific information on the topic `foo`. For more detailed information you can use the `info` command, e.g., `info foo`.

A Multi-tasking and Multi-user operating system

Unix is somewhat different from most typical personal computer operating systems. For one thing, multiple users can be logged in at once. Also, the computer can be running multiple tasks at the same time. A single CPU can only execute one instruction at a time, but it rapidly switches between tasks so they appear to be running simultaneously.

You can determine who's logged onto a system by typing `who`. You can see what processes are running (and what resources they are consuming) by typing `top`. Type `q` to quit the `top` process.

You should think of the Unix system as taking input from one or more “files” (which includes the keyboard, as it treats devices as a kind of file). This information can then be passed through one or more processes to produce an output file. Rather than having to store intermediate results in temporary files you can “pipe” the results from one process into the input stream of another process. This is one of the really powerful aspects of Unix, but it takes a while to embrace this perspective.

An example might help. The output of the `ls` command can be piped into the `more` command using the pipe symbol, which is a vertical bar:

```
ls -l | more
```

mini Unix Survival Guide

<code>cd <i>name</i></code>	changes directory to <i>name</i> (relative or absolute path)
<code>cd ..</code>	changes directory up one level
<code>cd ~</code>	changes to home directory
<code>cd</code>	(with no argument) changes to home directory
<code>pwd</code>	print current working directory (i.e. where am I?)
<code>ls</code>	lists contents of current directory
<code>ps</code>	lists processes running on the machine
<code>mkdir <i>name</i></code>	creates a directory called <i>name</i>
<code>rm -i <i>file</i></code>	deletes <i>file</i> , but asks first.
<code>rmdir <i>directory</i></code>	deletes directory, but it must be empty first
<code>more <i>file</i></code>	types the contents of <i>file</i> to the screen, one page at a time
<code>cp <i>file1 file2</i></code>	makes a copy of <i>file1</i> called <i>file2</i>
<code>> file</code>	sends output stream to file, overwriting if it exists
<code>>> file</code>	appends output stream to end of file
<code><i>task1</i> <i>task2</i></code>	pipes the output of <i>task1</i> into <i>task2</i>
<code>man <i>topic</i></code>	produces help listing on topic.
<code>info <i>topic</i></code>	produces help listing on topic

Troubleshooting

If you find that the cursor has gone away, with no indication of ever returning, try the old standby CTRL-c. This means hold down the Ctrl key and hit “c”. This ought to kill the hung process. If this fails, try CTRL-z, where you hit “z” instead of “c”. This suspends, rather than kills, the hung process but still might let you recover.