

ASTR 377
Dr. Chartas
August 23, 2016

Computer Setup

The computers used in this lab will primarily be Apple MacBooks running the MAC OS X operating system. Current Mac operating systems are built upon a UNIX platform, and it is this UNIX platform that we will be using. You will need to develop basic unix skills as well as mastering some programming (IDL). This exercise introduces you to the computing environment and teaches you how to get started in unix. Later exercises will introduce you to IDL and other useful tools.

I. HARDWARE SUMMARY

Apple MacBooks—running Mac OS-X, which is a unix-based operating system. The windowing system is so powerful that most of you weren't even aware that unix is "under the hood". It is this combination of both worlds that makes OS-X an appealing operating system for many observational astronomers. Each student will be assigned a specific laptop where they can create a folder on the Desktop to store any work related to ASTRO 377. Your folder should be backed-up often (at least after each lab) on a USB drive.

II. POLICIES

This lab is shared among many students and classes. Please be courteous to others by keeping the computer area clean. Eating is not allowed in the computer lab. If you must have a drink with you, please place it on the floor.

The laptops are to be used **only in the SSMB building or used when we go out observing**. Please shut-down your laptop before you place it back in the cart. You will be given access to the lab and laptops after class hours to work on your reports and projects.

III. The UNIX FILESYSTEM

You can think of the unix operating system as a single system of "files". Other operating systems are too, but using a windowing interface has hidden most of that from beginning users. You'll need to master the file system right away. You can think of it as a tree or as a system of folders and subfolders (and subsub and subsubsub ...). Folders (traditionally called directories) can contain either simple files or subdirectories. EVERYTHING is somewhere in

the file system – hardware, software, user files, networked devices; everything.

Where are you in the file system? First you will need to open a **terminal window**. Type **pwd** in your terminal window and carefully note the system response. What you see is called your "home" directory. Whenever you log in, this is where you start out. **pwd** means "print working directory". It's a typical unix command: short, obscure, and lower case. It probably responded something like `/Users/yourname`. `yourname` is the username you were assigned and used to log in. That's your identity on the system. Anything from this level of the file system on down, you control. Anything "above" this in the file hierarchy is something else.

Before learning how to move around, let's see what you have already. The command to view the files in your directory is **ls**. Type that and tell me what you see. Possibly nothing, because you haven't created anything yet. OS-X creates all sorts of junk when your account is created, so you'll probably see some file names. **ls** by itself doesn't tell you anything except the names of the files in the current working directory (i.e. the one **pwd** told you about). Like most unix commands, **ls** can be used in a whole bunch of ways. Here are a couple simple extensions of the **ls** command to try:

- **ls -a** some files are "hidden"; their names start with a `.` in order to see them, you need the `-a` option.
- **ls -l** this shows you a full listing of all the files and their properties.
- **ls M*** this lists everything that begins with the letter M. You can use this command in many different ways (e.g. **ls *100*** to display all filenames with "100" in it.

Let's look at the **ls -l** results. The first column on the left has a `-` if this is a regular file, a `d` if it is a directory, and one of several possible symbols if it's a special file. If you have a directory in there somewhere (let's say it's called "directoryname"), type **ls directoryname** to see what's inside it. The next 9 columns are very important. There are 3 kinds of file access, and there are 3 levels of file "ownership".

The first level is called "user". In this example, the user is you. The first 3 columns list whether you are allowed to read (r), write (w), or execute (x) to the files. The only things that should be executable are compiled programs and directories. An executable directory means that you can "move" into that directory and see what's in there and read and write files in there. If you have write access, you also can delete the file. The default is `rw` for directories and `rw-` for normal files.

The second level is called "group". Groups are assigned by the system administrator. You have a unique username, but you may be a member of many different groups. Student accounts are all in the "students" group, for example. Different research groups might have their own group name (e.g. "grb"). The next 3 columns show what anyone who is in the group can do to your files. The default is r-x for directories and r— for normal files.

The last three columns describe what anyone who is logged into the system ("other"), or anyone who has access to the file system (even if they are not logged in to this computer) can do with your file.

Later we'll learn how you can control these attributes. You should never create files that give "other" access to modifying your files (e.g. rwx and rw- are not ok). But unix is a collaborative system, and there are many good reasons we might want to share files. So by default, your "home" directory is rwxr-xr-x. Therefore any files in your home directory can be viewed, but not changed or deleted, by anyone on the system. You will learn how to create directories that can only be shared with members of the group or with nobody (except the notorious "superuser").

After the access control, you'll see a column telling you who the owner of the file is. The next column tells you the group ownership of the file. Then there's a column telling you how big the file is. This number can take several forms. Generally for normal files, it is the number of kilobytes. Big numbers mean big files.

The next column gives the date and time that the file was last changed. Finally, the last column gives you the full file name.

A few words about file naming conventions. Unix is case sensitive. File1.txt is not the same as file1.txt. Extensions are useful to give a sense of what's in the file, and they are used by some applications (like the finder on a Mac) to associate the file with an application. But they are not special in anyway to unix. Any of the following are valid filenames: file1.doc file1_aworddoc file1 file1.doc.mydoc.quack.quack. It is ALWAYS a bad idea to include spaces in filenames. The problem is that unix commands are separated by spaces, so unix often gets confused if you are doing something with a file and you encounter a space. So don't use them in filenames. If your filename starts with a period (e.g. the file .cshrc), it is a special "hidden" file. That's really the only thing that is special about it: you can't see it unless you put a -a option on the end of your ls command. They are often used to set up preferences for programs and applications, and they are hidden for your convenience only. Your home directory will fill up with them as you progress. Another character you must never use in a unix file name is the / character. Can you see why? Type **pwd** again... / plays a special role in the unix file system.

For example, your home directory might have been /Users/yourname. That means there's one level above your home directory, and it's called /Users. There's even a level above /Users. What do you think it's called? Let's find out. Type **cd** and then **pwd** to make sure you are starting in your home directory. Now type **cd ..** then **pwd**. **cd** ("change directory") with no arguments returns you to your home directory from anywhere on the system. The **..** argument to **cd** means "move up one level". You should now be in /home.

Now let's look inside a important file called .cshrc. Type **cd** to go to your home directory. The file called .cshrc is special; you'll use it to define a bunch of defaults, system variables, paths and aliases. To see what's inside, type **less .cshrc**. Scroll up and down and then type **q** to quit. Let's go home now. Type **cd** again and then **pwd** to make sure. Another way to get to your home directory would be to use its full pathname. Type **cd /Users/your_username** then **ls** and you should see the same thing you saw a minute ago. In other words, you move around using **cd**, and you can either give an "absolute path" starting with a / and including everything above that directory, or you can give a "relative path" (from here up, from here down, etc). The absolute top level directory is called the "root directory". Change directories until you are there and type **ls -l**. Everything will be owned by funny names like "root" and "admin", which means you can't change it. But you have read access to alot of it, and you have execute access to many of the directories, so you can look around and see what's where. Practice for 5 or 10 minutes or so snooping around the system with **cd**, **ls**, and **less**. When you are done, return to your home directory. To learn more details about a unix command type **man** followed by the name of the command. Try typing **man ls**

Some other useful unix commands:

cp file1 file2

This command makes a copy of file1 and names it file2.

mv file1 file2

This is command deletes file1 and creates an identical file named file2.

Be careful using this command!

mkdir dirname

A command to make a directory name dirname

history

A command that shows what commands you have recently typed in the terminal window

top

display and update sorted information about processes

rmdir dirname

removes the (empty) directory with name `dirname'
Be careful using this command!

rmdir -r dirname

recursively removes directories and subdirectories
Be careful using this command!

command > file1

redirects the output of 'command' to 'file1' instead of to standard output (screen)

command >> file1

appends the output of 'command' to 'file1' instead of to standard output (screen)

command1 | command2

pipe standard output of command1 to standard input of command2

more file1

displays the file 'file1' a page at a time

ctrl C

interrupts whatever is currently running.

grep word files

search inside files for word

The Mac laptops, will be used for most of your research including running hardware such as the telescope digital cameras.

You will need to be comfortable with the UNIX operating systems as well as master the IDL programming language.

Commonly Used Programs for ASTRO 377

Make sure you can access the following software tools from the terminal window: **idl**, **ds9**, **fv**, **xspec**.

fv and **xspec** are part of a much larger package called HEASOFT that is installed on your laptops. To learn more about all the tools available to you in this package see

<https://heasarc.gsfc.nasa.gov/docs/software.html>

Open a terminal window and type **idl**. You should see the prompt IDL>
To exit **idl** type **exit**. The IDL Astronomy User's Library is installed on your laptop. To see one-line descriptions of the IDL procedures go to:

<http://idlastro.gsfc.nasa.gov/contents.html>

Next type **ds9** in the unix shell (not in **idl**). The **ds9** GUI should appear. If **ds9** is not installed on your laptop find it online using google and install it on your laptop. Download the file **ccd7.fits** from OAKS.
Open the fits file **ccd7.fits** using **ds9**.

Try to make yourself familiar with some of the features of **ds9**. What object do you think was the focus of this observation (hint look in the header of the file). Try the various

Exit **ds9** and see if you can access the fits viewer tool called **fv**.

Type:
fv ccd7.fits &

Try to make yourself familiar with some of the features of **fv**.

Text Editor

Emacs is a very powerful multi-windowing text editor.

For some useful commands go to:

<http://mcsp.wartburg.edu/ucl/emacs.html>

You can use emacs to write IDL software. First download from OAKS the fits file `lcbx01010_x1dsum.fits`

Example: In a terminal window type **emacs plotspec.pro**.

Then type in the following:

```
pro plotspec
; Program to open and display a spectrum
;
;
file='lcbx01010_x1dsum.fits'
spec = mrdfits(file,1)
flux = spec.flux
wav = spec.WAVELENGTH
err = spec.error
bkg = spec.BACKGROUND
exp = spec.exptime
plot_io,wav,flux,yrange=[1.e-16,5.e-13],xrange=[1150,1300]
end
```

To save the above graph as a postscript file type the following commands:

```
IDL>set_plot,'ps'
IDL>device,filename='spectrum.ps'
IDL>.r plotspec
IDL>plotspec
IDL>device,/close_file
IDL>exit
```

Assignment 1

1.1

Now write your own IDL program that plots the illuminated fraction of the Moon for one month starting from today . Hint, search the IDL Astro Library for a related program called `mphase.pro`. Use **more** to look at the example in the description provided for `mphase.pro`. Label axes.

Submit your program and a text file called `readme.txt` to DROPBOX that includes your name, the date, and a brief description of this program and relevant information for this question and the one that follows.

1.2

Save the postscript file of the plot you made of `lcbx01010_x1dsum.fits`. Label the X and Y axes with appropriate units.

- By checking the header of the fits file (hint use `fv`) report the name of the object and the telescope used to observe it.
- By using **NED** (<https://ned.ipac.caltech.edu>) determine the nature of this object and its redshift.
- Where does the emission and absorption line flux come from?
- Where does the continuum emission originate from?
- By using the `linetable`

<http://classic.sdss.org/dr6/algorithms/linestable.html>

identify the broad emission line in your spectrum.

Submit your postscript graphs and idl programs to OAKs DROPBOX and include in a file named `readme.txt` a description of what you have plotted and your answers to Assignment 1.

