

Astronomy 480 || Linux Tutorial I

We'll be using an operating system called "Linux" on the computers used for this course. Linux is a form of UNIX, a multi-user multi-tasking operating system. For the purposes of this class, we won't delve deeply into the details of the operating system, but you should feel free to take the opportunity to acquire a deeper working knowledge of Linux as the quarter progresses. [One of the O'Reilly books, "Running Linux," has just about everything you need to know.]

Summary for our machines

The machines in PAB B356 run Red Hat Linux. There are 18 machines, given the identifications from astrolab01 to astrolab18. All home directories are hosted on a common disk farm, so you always land in the same home directory, but running on the processor you logged on to. Note that you can access these machines and their contents only via encrypted ssh and scp (secure shell and secure copy). We'll learn more about that shortly.

Logging In

The first step in using the computers at our disposal is to log in. If you pre-registered for the class, an account has already been established for you. The username on the account is (in general) the same as the name you use to receive email at **u.washington.edu**. The initial password for logging into your account will be given to you in class.



The first step in today's exercise is to log in and then change your password.

- Click on **>Session** at the bottom of the screen even before logging in. Make sure that "default system session" is marked so that we are all operating under the same rules.
- Log onto your console. Type your username and then a carriage return, followed by the password that was given to you by your instructor (if you didn't already have an account).
- You will be dropped into the "KDE" windows manager, under the Linux operating system.

Customizing the tool bar

Along the bottom tool bar you'll find a number of icons. If you see no icons along the gray strip at the bottom, or not the ones you think you may want, then follow the next series of steps.

- right click on the grey tool bar
- click on **Add** ►
 - Application button ►
 - Graphics ►
 - PDF Viewer
 - Internet ►
 - Firefox
 - Programming ►
 - Emacs Text Editor
 - System Tools ►
 - Terminal

A good thing to know about is the “Control Center” where you can personalize nearly everything on your computer. Another way to do the above is to click on the  “K” symbol or the  redhat “RedHat” symbol and a menu should pop up. Explore and find what you need.

Open up a terminal and change your password!

Computer security is an issue of ever-increasing importance. The UW Astronomy Department is under a constant barrage of attacks from people who presumably think we have the truth about alien abductions, cold fusion, etc. This is more than a nuisance. Our ability to conduct our teaching and research programs depends upon reliable and secure connections between our system and computers across the globe. You are expected to help us keep our shared systems stable and secure.

Individuals who compromise system integrity through irresponsible sloppiness will have their computer accounts deleted.

Linux passwords should be at least 8 characters in length, and it is essential that you not use any words that appear in a dictionary, or simple substitutions of one or two letters in a word. An ideal password is a combination of characters and punctuation symbols, with some upper case letters thrown in for good measure.





Use the command `passwd` to change your password. It will prompt you for the old and new passwords, and will not echo what you type. You'll be asked to confirm the new password a second time, to prevent inadvertent typos.

Logging out

When the time comes, you can log out by right-clicking on the screen or left-clicking on the “K” or “Red Hat.”

Window Management

Linux operates in two ways: 1) via window management much like Windows or Macintosh and 2) via command lines (which we will be emphasizing). We will need to be proficient in both ways. We'll start with window management, but within a terminal (Shell-Konsole) or an `xterm`, we will be using command lines.

Figure out how to move, resize, iconize, and kill the shell window you opened up; it is similar to Windows. Move the window around by dragging the top bar. Try double-left-clicking on the top blue bar. What happens? Use the mouse to drag the lower corners of the window to resize it. Clicking on the — the  or the  brings the expected results (maximize, kill), as should the  once you clicked on . A minimized window still appears on the tool bar. Is there an icon in the upper left-hand corner of the terminal? What does clicking on it do? How about the lower left-hand corner?

Open up a second window, this time an `xterm`, by typing

```
xterm & ; the & means run the process in the background
```

at the prompt. It should open up a new window that will also accept commands. You may note that these two windows have some subtle (and important for us) differences. We get into this later. You can kill the window by typing `exit`. For experimenting, type in just `xterm` without the `&` and see what happens. This is a good way to tie up two terminal windows at once, not recommended.

Customize the Window Manager – Important for IRAF

The next task is to customize the way the mouse interacts with the active windows on your desktop. We need to change the default setting in order to work efficiently within IRAF. What happens will sometimes be frustrating, but it is necessary, so here goes!

First, open up two active shell windows (terminals or xterms) on the desktop if they are not already open. For the programs we'll be using, we want the position of the mouse to determine which window is active, not any mouse clicks.

Right click on the symbol at the upper left corner of a terminal. Click on “Configure Window Behavior” and on “Focus.” Under “Policy,” choose “Focus Strictly Under Mouse,” and “Auto raise.” Choose a delay if you'd like (recommended). Test this by hitting some carriage returns with the mouse over one window, then position it over the second one and hit a few more carriage returns. The active window should be determined by the mouse location, without your having to click to activate the window.

Investigating Linux Commands

Directory Structure

The directory architecture used by Linux is a hierarchical tree structure, much like what you're accustomed to in a Windows or Mac environment. When you first log onto a Linux system, you land in your "home" directory. You can navigate through the directory structure by "changing directories", using the command `cd`.

(Note that Linux is **case sensitive**, i.e. `CD` won't work, it has to be lower case.)

To indicate the present working directory, you can type `pwd` (print working directory). The prompt as currently set up on your machine may or may not list the directory you are working in.

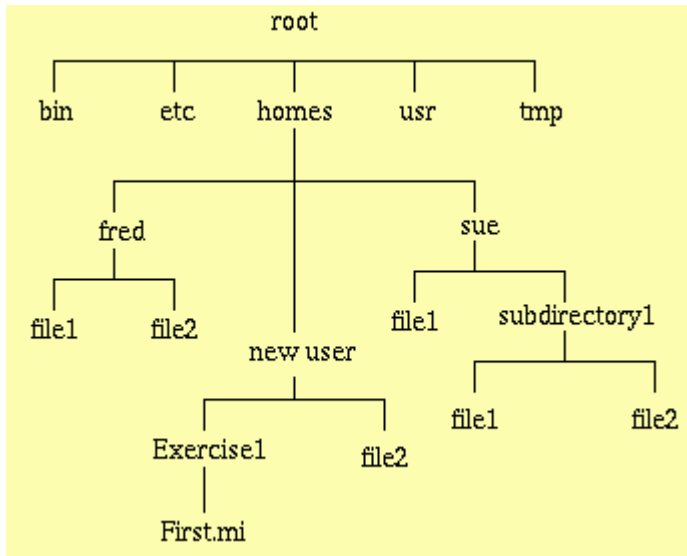
Figure 1 below shows an example of a portion of a Linux directory structure. The top of the tree is the "root" directory, designated with a slash sign. Directory names build on the root directory, with a slash designating a new branch in the directory structure. The direction of the slash sign is the opposite of what is used in Windows/DOS systems. For example, the full (absolute) path name for your 'home' directory would be `/astro/users/yourusername`. The absolute path to your course directory is `/net/projects/Astro_480/spring-06/yourusername`. [We'll learn how to make a shortcut to your course directory so that you don't need to type in the whole thing every time.]

NOTE: You should use your home directory for “everyday” files, not for research or course studies that will take up a lot of disk space like this course will. We will investigate directory structure below. Although some of you may have been working in `/astro/users/yourusername/`, and may, in fact, be taking up a lot of disk space because of it, your directory for your files and work for this course is:

`/net/projects/Astro_480/spring-06/yourusername/`

This is Critical: Never, ever just turn off the machine in order to "reboot". Linux does not tolerate that very well, and will likely foul up the local disk. Recovery from that is a painful process. Let an instructor know if your system is frozen. Chances are very good we can fix it by logging in from a different machine. **Security Note:** Never, ever, ever, ever use telnet or ftp to access the astronomy department computing system. Instead use ssh and scp, which are the secure, encrypted equivalents of these utilities.

Figure 1. A Linux Top-Down Directory Tree.



Moving around the paths

For now, you need to be working in your home directory. For all of the Linux practice (and, really, up until we start using IRAF), working in your home directory will be fine. You will be creating many, many large image files once we get into IRAF, however, and will need to be on a different disk.

Linux commands require a space after them. Type in the following:

```
cd ;Where are you? Remember pwd ??
cd ../ ;Step up 1 directory. Where are you? [Note: there is a space after the 'cd']
cd / ;Where are you? [Space after 'cd'] What's in this directory?
```

Return to your home directory.

What is the difference between a 'relative' and an 'absolute' path? You were just working in a relative sense, choosing your path based on where you are, relatively speaking. If you type

```
cd /net/projects/Astro_480/spring-06/
```

you are working in the absolute sense, from the root directory. If you wanted to remain relative, you could also accomplish the same thing by doing the following (starting out in your home directory):

```
cd ../../ ; You should be in the root directory
cd net
cd projects
cd Astro_480
cd spring-06
```

The UNIX Command Prompt

The UNIX command prompt, or shell, as it's also called, provides you with direct access to the operating system. It is a command interpreter: It interprets and executes all of your commands. The shell is really just another program that sits between your keyboard, the operating system, and other programs. Unlike DOS, the shell is a separate program from the operating system, or kernel.

Though powerful, shells are lacking in the ergonomics department: They just don't provide a friendly and intuitive GUI. And no matter how ergonomic the shell, the fact remains it is a shell, and it wants you to type in commands.

Yes, shells are by nature arcane and typing intensive, but they also are powerful. Shells provide many options unavailable to GUI-only systems. When you learn how to use a shell, you won't be able to do without one. Granted, most things are easier to do with a GUI application, but many other things are not. The shell is then the right interface for certain kinds of problems.

If you are accessing a UNIX box remotely, you will more than likely do so through a shell interface. Shells come in different flavors and knowing how to use a shell is a good thing. It is a wise person who is familiar with at least two different shells. It will be of tremendous help if you ever have shell access to a system that doesn't sport your favorite shell.

Most shells in UNIX are programmable, so if you find yourself writing the same commands over and over, it is easy to create a shell program (similar to a .bat file in DOS and Windows) that automates those commands for you. But you can worry about this when the time comes.

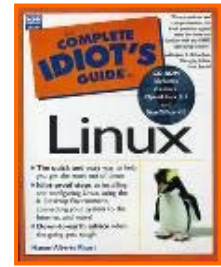
Command-Line Program Syntax

Most command-line programs use the following syntax:

```
commandname [flags] arg1 arg2 argn
```

The terms are defined as follows:

- ◆ `commandname` is the name of the command.
- ◆ `flags` or arguments are options given after the program name that control how the program will behave. Flags are typically differentiated from other types of argument by a leading dash or double dash, as in `-h` or `--help`. Other types of arguments can be filenames or some other program-defined keyword. Flags are case sensitive, and tools with many options will use both lowercase and uppercase letters to name their options.
- ◆ `arg1-argn` is a list of arguments for the program to work with. Typically this will be a list of files, the name of a directory, or something else required by the program to do its thing.



What shell are you running, command-line program syntax, and more good stuff.

What shell are you running?

```
echo $shell      which part is the "command"?
```

Which part is the argument?

Are there any flags? Could there be? To answer this, type:

```
man echo and see what happens.
```

Typing `man` followed by the `command name` gets you to the help manuals.

Listing files

What do you have in your current directory? What folders (sub-directories)? What files? Are there any hidden files? How do you find out?

Type:

```
ls
```

What does this command give you? Let's add some arguments.

```
ls -F
```

How many directories do you have in your home directory? How many files? Try:

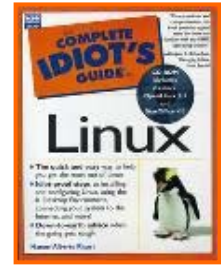
```
ls -alt
```

Any hidden files? How many? Name one. The `-a` reveals the hidden files; `-l` gives the complete file name and all of the permissions; the `-t` gives them in the order of most recently accessed.

Table 12.1 Useful ls Options

Option Action

- F Mark files with characters designating their type
- l Display long listings
- a List all files including any hidden files
- s Print the size of files
- t Sort listing by file time, with new files first
- R List files recursively including contents of other directories
- I List file inode



You can combine multiple flags together, as in `ls -Flai`. This is the same as saying: `ls -F -l -a -i`, except that you save a few keystrokes.

example:

```
[alberto@digital alberto]$ ls -Flai
total 9
157289 drwxrwxr-x  9 alberto  alberto   1024 Sep  1 10:25 ./
169675 drwxrwxr-x 16 alberto  alberto   1024 Sep  1 10:24 ../
157290 drwxrwxr-x  2 alberto  alberto   1024 Sep  1 10:24 Desktop/
157291 drwxrwxr-x  2 alberto  alberto   1024 Sep  1 10:24 Mail/
157292 drwxrwxr-x  2 alberto  alberto   1024 Sep  1 10:24 html/
157293 drwxrwxr-x  2 alberto  alberto   1024 Sep  1 10:24 images/
157294 drwxrwxr-x  2 alberto  alberto   1024 Sep  1 10:24 kde/
157295 drwxrwxr-x  2 alberto  alberto   1024 Sep  1 10:24 lg/
```

For the sake of being complete, the items listed in the listing are as follows (from left to right):

- The inode of the file
- The permissions for the file
- The number of hard links to the file
- The owner of the file
- The group owning the file
- The size of the file
- The last modification date for the file
- The name of the file

For the folders and files in your home directory:

- who owns the files?
- when were the files or directories modified?

d rwxrwxr-x	2	alberto	alberto	1024	Sep 1 10:24	kde/
type of file (directory) + permissions; 3 for owner, group, other to <u>r</u> ead, <u>w</u> rite, <u>e</u> xecute	# of hard links	owner	group	size in bytes	date and time of last modification	name

Try the `ls` in technicolor, and comment on what you get:

```
ls --color
```

Making Shell Options the Default

Change directories

```
cd ENV
```

so that you are in the ENV subdirectory. Get a list of the hidden files. Here's a jump ahead on how to find out what 'aliases' have already been set up for you as a natural part of creating new directories. Type the following:

```
grep alias .*
```

['grep' is the command; you are looking for the word 'alias' in all hidden files (.*) in this directory.]

Take a look at one of these files by typing in:

```
less filename
```

Type "q" to quit 'less'

Here are some examples of what I have in my /ENV/.cshrc.personal file ('cause I'm an impatient typist as well....) The # sign tells Linux to ignore whatever comes after it.

```
##### Define any other aliases for commands here #####
alias ls      'ls -F --color'

alias cp      'cp -i'           # ask if about to write over something
alias mv      'mv -i'           # ask if about to write over something
alias rm      'rm -i'           #ask if about to delete!

alias a101    'cd /net/www/larson/Astro101'
alias a150    'cd /net/www/larson/Astro150b'
alias a421    'cd /net/www/astro421/'
alias a480    'cd /net/www/astro480/'
alias a481    'cd /net/www/astro481/'
alias observatory 'cd /net/www/observatory/'
alias ssynth  'cd /net/projects/SSynth/'
```

We will be learning two of the editors in Linux – emacs and VI – in the next class period. You will find out how to modify these files to suit your fancy (preferences).

Shortcuts

The Unix command language was put together by someone who hated typing and who wanted to put in as few keystrokes as possible. Unix, and thus Linux, has a huge number of shortcuts. Here are a few:

Finish a file or directory name by typing just enough letters that make the name unique and then hitting the **TAB** key.

Use the "up-arrow" key to back up your list of recent commands. Hit return when you get to one you want to use.

Type:

`history` ;What do you get.

Type in

`!10` ;(or some other number from that list) and see what happens.

Viewing files, more or less*

You can invoke an editor, such as:

`emacs .cshrc.personal` ;or, obviously, any filename you want

You can scan the file quickly:

`cat .cshrc.personal`

A long file will literally fly by if you use 'cat' (but `cat` can be a handy way to concatenate files).

Use:

`more .cshrc.personal`

to take your time. Contrary to logical thinking, another command, `less`, is actually more powerful.

Try:

`less .cshrc.personal`

With `less` you can page up or page down, search down for a pattern by using `/ pattern` or search up by using `? pattern`. (But, without the quotes.)

Stuck?

`Ctrl-c` will abort a process

`Ctrl-z` will suspend it

Computer frozen? Try `Ctrl alt F1` and then `Ctrl alt del`

To log yourself off and return to the log-in window, right-click on the screen, or left-click on the "K" icon.

Congratulations. You've made it through the initial start in learning Linux. More fun coming up!

* Welsh, Dalheimer, Kaufman (1999), *Running Linux* (O'Reilly & Associates, Inc.), p. 97