

Astronomy 480 || Linux Tutorial II

A brief review

Changing directories using command lines in a terminal

You can tell where you are in the directory structure by typing `pwd`

Here is part of what I have in my directory `/astro/users/larson/` [directories are shown in bold]

Figure 1

```
[larson@astrolab18 ~]$ pwd
```

```
/users/larson
```

```
[larson@astrolab18 ~]$ ls
```

1979PASP...91..589B.pdf.gz	PG1528+062.ps	login.cl.OLD
30536-25.JPG	celestial_mechanics	m13CMD.ps
5085.pdf	celm13.pdf	m57c001.FIT
Astro480	clusters_dao_stromgren.sxc	mail
BV.ps	daophot2.ps	manuals

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. If I were already in the "larson" directory shown in Fig. 1 above, and I wanted to move into the directory called "Astro480" I would simply type:

```
cd Astro480
```

and the system would move my current working directory to `/users/larson/Astro480/`. If I wanted to move to this directory from somewhere other than the directory right above it (also called the parent directory) I could use an absolute pathname by typing:

```
cd /users/larson/Astro480
```

Linux provides some useful shortcuts for navigating directories. A single dot, ".", always refers to the current directory. Double dots, ".." refer to the parent directory. To move up two levels in the directory structure, you would type:

```
cd ../../
```

The tilde symbol, "~", is a shorthand for home directories. If you wanted to move to my home directory, from anywhere in the directory structure, you would type:

```
cd ~larson
```

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

Listing the contents of a directory using command lines

The command `ls` is used to list the contents of a directory. Under Linux, files, directories and even devices (such as the CD drive) 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 Linux, command options are preceded by a hyphen. A few of the more useful forms of `ls` include:

`ls -l` generates a full listing, including dates, sizes, etc.

`ls -p` distinguishes between directories and regular files

`ls -a` lists all files, including those that begin with a ".", which are otherwise hidden files.

`ls -t` lists contents by date of their creation, latest first.

You can combine them, so for example `ls -apt` is perfectly valid.

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

```
/users/larson/foo.txt
```

where the last element, `foo.txt`, is the file name.

Linux allows you to select a subset of the objects of interest, using wildcards and range restrictions. For example, to obtain a listing of all files that end in ".doc", you would type:

```
ls *.doc
```

Similarly, the character "?" in a filename allows for any *single* character to occupy that position in the file name. If I had a list of successive astronomical images called **obj.001.fits** through **obj.120.fits** and I wanted only those in the range 120-129, I could use:

```
ls obj.12?.fits
```

You can also specify a range of characters and numbers, by enclosing the range in square brackets. To get a list of all images with numbers between 111 and 119 I could use:

```
ls obj.1[11-19].fits
```

Copying Files: cp

To copy a file, use the `cp` command. The `cp` has the following syntax:

```
cp [options] source destination
```

`source` is the name of the file you want to copy, and `destination` is the name of the copy you create.

If `destination` is a directory (the path didn't specify a filename), the file is copied with its original name into the specified directory.

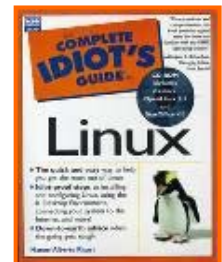
The `options` argument is shown between brackets to say that `options` are optional. We'll use an option on the `cp` command in the next section.

Here's an example:

```
[alberto@digital alberto]$ ls -F
Xrootenv.0  afile          anotherlink  helloworld*
adir/      alink@        hello

[alberto@digital alberto]$ cp helloworld /tmp

[alberto@digital alberto]$ ls -F /tmp
fvwmrca00455 helloworld*  install.log  screens/
```



You can copy multiple files to a single directory in a single command by listing several source files separated by spaces. The only requirement is that the last argument (destination) be a directory.

Copying Directories

You copy directories in much the same way you copy files. When copying directories you need to tell `cp` that you want to copy recursively all the files and directories contained by the source directory. This is easily done with the `-R` option. The syntax looks like this:

```
[alberto@digital alberto]$ ls
Xrootenv.0  adir/          afile          alink@         anotherlink
helloworld*
[alberto@digital alberto]$ cp -R adir /tmp
[alberto@digital alberto]$ ls /tmp
adir/          install.log
```

Moving Files and Directories: `mv`

To move a file or directory, you use the `mv` command. Moving files is similar to copying files. The big difference is that the original file is removed after the new copy is created. The `mv` command follows this syntax:

```
mv [options] source destination
[alberto@digital alberto]$ ls
Xrootenv.0  adir/          afile          alink@         anotherlink
helloworld*
[alberto@digital alberto]$ mv afile file2
[alberto@digital alberto]$ ls
Xrootenv.0  adir/          alink@         anotherlink file2
helloworld*
```

The `mv` command can also be used to move an entire directory tree.

The `mv` command provides several options. For more information, refer to its manual page.

Copying Files

The command `cp` is used to copy files. The syntax of this command conforms to the (almost) general UNIX standard of:

`command source destination`

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

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

Creating Directories

You can create a new directory just below the current working directory by using the command `mkdir name`, where `name` is the name of the directory you want to create. Avoid spaces in directory names, Linux hates spaces in directory and file names and lets you know it!

Directories are deleted with the `rmdir` command. The directories will need to be empty before they can be removed. Feeling vicious? Try this on a directory where you do not want what's in it:

```
\rm -R directoryname/*  
rmdir directoryname
```

The backslash in `\rm` tells Linux you don't want confirmation before removing files, and, in this case, subdirectories and everything in them. Once gone, files and directories cannot be retrieved unless they were backed-up. There's no menu that you can go to that says "undelete." Trust me, I know.

Viewing the Contents of a File

The commands `cat file` and `more file` print the contents of a simple text file to the screen. The `cat` command scrolls the entire thing past with no pauses. The command `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. `Less` is even more flexible as you can search that file and move up and down, mark your place, plus more. You should type the following:

```
man less
```

and read what it says. By the way, if you need help on some topic and do not know the exact command name, or command that involves what you want to do, use:

```
man -k whatIwanttoknowabout
```

Try:

```
man image
```

and then:

```
man -k image
```

and see what we mean.

Here are a couple more handy commands (see **man** pages for available flags):

◆ **head**-Peek at the first few lines in a file.

◆ **tail**-Peek at the last few lines in a file.

Redirection & Saving Your Output

One of the most useful aspects of Linux 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:

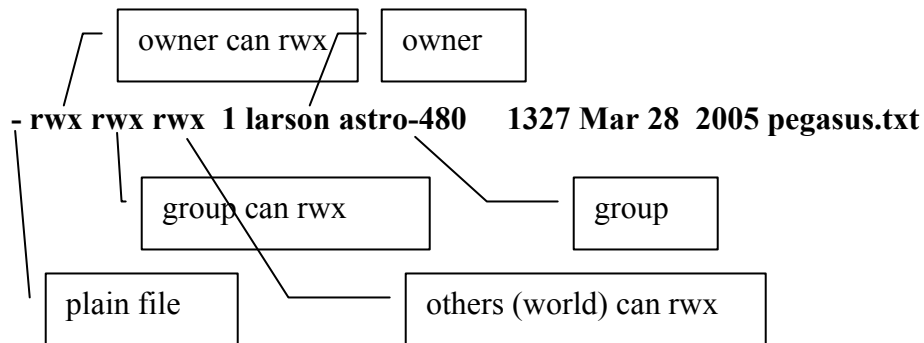
```
ls -a > foo
```

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 -a >> foo
```

File Ownership and Permissions

Let's return for a moment to this idea of owning a directory or a file and being able to set permissions for others.



What if you wanted others to be able to read your files but not be able to write, change, or execute any of them? Take a look at this listing from `/net/projects/Astro_480/spring-06/larson/exercise1`

```
-rw- r-- r-- 1 larson astro-480 1546126 Feb 27 18:42 Vflat.0001.fits.gz
```

It is a file, the owner can read and write but not execute (but it's not executable anyway), the group can only read the file (or copy it to their directory, but not move nor delete), the world can only read the file. There may be cases where you cannot get a file I have requested you get from one of my directories. I can fix that in a jiffy with the command **chmod**. Do the following in your directory:

```
[larson@astrolab18 test]$ touch trialfile ;create an empty file by 'touch'
[larson@astrolab18 test]$ ls -l
-rwxrwxrwx 1 larson astro-480 73 Apr 4 2005 bill
-rw-rw-rw- 1 larson astro-480 954164 Apr 4 2005 out.txt
-rw-rw-r-- 1 larson astro-480 0 Mar 28 18:16 trialfile ;default permissions
[larson@astrolab18 test]$ chmod ug-w trialfile ;prevent writes by everyone, even you
[larson@astrolab18 test]$ ls -l
-rwxrwxrwx 1 larson astro-480 73 Apr 4 2005 bill
-rw-rw-rw- 1 larson astro-480 954164 Apr 4 2005 out.txt
-r--r--r-- 1 larson astro-480 0 Mar 28 18:16 trialfile
[larson@astrolab18 test]$ chmod ug+x trialfile ;make it executable
[larson@astrolab18 test]$ ls -l
-rwxrwxrwx 1 larson astro-480 73 Apr 4 2005 bill
-rw-rw-rw- 1 larson astro-480 954164 Apr 4 2005 out.txt
-r-xr-xr-- 1 larson astro-480 0 Mar 28 18:16 trialfile
[larson@astrolab18 test]$ chmod ug+w trialfile ;add write back in
[larson@astrolab18 test]$ ls -l
-rwxrwxrwx 1 larson astro-480 73 Apr 4 2005 bill
-rw-rw-rw- 1 larson astro-480 954164 Apr 4 2005 out.txt
-rwxrwxr-- 1 larson astro-480 0 Mar 28 18:16 trialfile
```

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. Then, you have to get rid of the hung process, but we'll figure that out later.

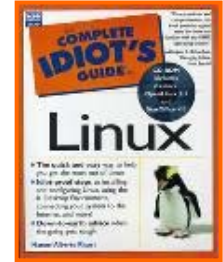
Remember, never just turn a Linux machine off with the on-off button, or you truly will be an idiot.

Filename Expansion: Tab

As I said earlier, shells are not very ergonomic, and most people don't like to type commands or long pathnames very often. Depending on the shell of you use, you might have the ability to have the shell complete filenames and command names for you. The shell will do this whenever the portions of the path you specified cannot be confused for something else.

To activate completion on `bash`, all you need to do is press **Tab** after you've typed enough characters. If you wanted a listing of the `/usr/local/bin` directory, you could type `ls /u<Tab>/lo<Tab>/b<Tab>`, and the shell will expand each of the paths for you, saving you some keystrokes.

If the shell cannot complete the expansion because multiple completions are possible, hitting **Tab** again will list everything that could be a match. This is very useful if you have forgotten the name of a command. For example if you type `l<Tab><Tab>`, the shell will respond with a list of all the programs that begin with an `l`.



More Working with Files and Folders

Creating some necessary directories

You will now need to go to your course directory and use the `mkdir` command to set up a sub-directory for your first exercise in IRAF: `exercisel`

That will be enough to get us started. Although we will not be starting our IRAF work just yet, we can set up the directory for the first exercise. Change directories so that you are in your 'exercisel' directory. Copy the necessary files from

`/net/projects/Astro_480/spring-06/larson/exercisel`

by using a relative path:

```
cp ../larson/exercisel/*.gz . ;there is a space after gz
```

[If this path doesn't work, figure out why not!] How many parts of the command do you recognize? What you have told the computer is to "go up one directory, over to larson, down to exercisel, grab everything that ends in .gz and copy it to the directory I am currently in exactly as everything was where it is being copied from (that's what the period at the end does)." Whew.

See what is now in your directory. These files are compressed, or 'zipped' (gzipped to be precise). We will leave them in that state for now.

Reading files

You've already been introduced to a bit of this. Copy the following file into a directory (preferably not the 'exercise1' one you just created) : /users/larson/pegasus.txt

Do the following things to that file:

```
cat pegasus.txt
less pegasus.txt
head pegasus.txt
head -10 pegasus.txt
tail pegasus.txt
tail -5 pegasus.txt
cat pegasus.txt > temp
cat pegasus.txt >> temp
cat -n pegasus.txt
```

[OK, are you using the tab key to shorten the number of strokes you use? Are you using the up-arrow key when you repeat all or part of a command?]

Text Editing Under a Shell

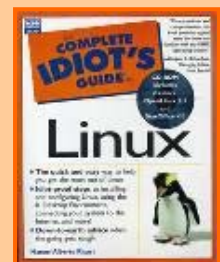
Vi (now known as *vim*) will be popping up unexpectedly for us in IRAF. You need to be able to recognize that that is what is happening and know how to edit a file within vi. Some may think that "vi" stands for very idiotic, but it has its time and place as an editor. It is fast, no new windows pop open, and only a few keystrokes get the job done. But, it is definitely not very sophisticated. We will work through a short tutorial here first. Then, we will move onto a more powerful editor called Emacs.

Vim Basics

Vim is a modal editor. All this means is that when you are using vim, the program behaves differently depending on whether you are in command mode or editing mode.

Let's go. Edit the Pegasus.txt file with vim. On your command line, issue the following:

```
vim Pegasus.txt
~
~
~
~
~
"pegasus.txt" 21L, 1327C
```



On issuing the command, the editor takes over your terminal. Empty lines are marked by a tilde character. [The tildes are the most obvious evidence that you are in vi; the other is that you won't be able to change anything until you get into the edit mode.] Vim is currently in its command mode. It is waiting for you to tell it what to do.

When in command mode, you can move around the document (assuming there's some text to navigate, which won't be the case in a new document) and issue commands. Table 13.1 lists all the cursor moving commands. (As you can see, some are mapped to special keys on your keyboard.)

Table 13.1 Moving Around the Document

Key	Function
l or Right Arrow	Move forward to the next character
h or Left Arrow	Move back one character
j or Down Arrow	Move down one line
k or Up Arrow	Move up one line
w	Move one word forward
b	Move one word backward

In general, you move your cursor by using the arrow keys (or their **hjk~~l~~** counter-parts).

[You can move quickly around the document by using [ir { or] or }. You can search for a string by using /*searchstring* to search down or ?*searchstring* to search up.]

The cursor will not wrap or move beyond the beginning or end of a line. When you reach the end, the computer beeps. To go to the next or previous line, use the down-or up-arrow keys, respectively.

When you begin a new, blank document, you don't have much to do; there's no text to edit or navigate. To add text, you need to get into the insert mode. This mode allows you to enter text as you normally do in other programs; you are still able to use the arrow keys to move around. (The **hjk~~l~~** keys type a letter when you are in insert mode.) Type **i** to get into the insert mode. Table 14.2 lists a number of editing commands that you can use while in command mode.

Table 14.2 Vi (Vim) Editing Commands

Command	Description
i	Insert text (begin editing)
x	Delete the character under the cursor (command mode)
dw	Delete characters until the end of the word
d\$	Delete text until the end of the line
dd	Delete the current line
u	Undo the last command
U	Undo any changes to the line
Ctrl+R	Redo any undos
Esc	Return to command mode, cancel current action-repeat until vim beeps

To return to the command mode, hold down the **Esc** key until vim beeps at you. Then you can type a **:+command**. Commands are issued by pressing the colon followed by the name of the command. Table 14.3 lists the basic ones to get you out of vim.

Table 14.3 Vi (Vim) Exiting Commands

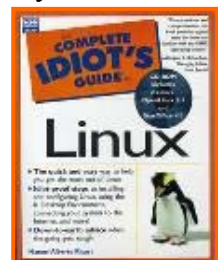
Command	Description
:q!	Quit without saving
:wq	Save the file and quit
:w <i>filename</i>	Save the buffer to <i>filename</i>

Vim has some editing commands, and the all-important **undo**. These editing commands work while vim is in command mode.

Editing with Emacs

From the Idiot's Guide: "Many hardcore UNIX people use only one program: Emacs. They start it when they start their work session, and they close it when they exit their system. All their work, with rare exceptions, is handled through Emacs, including access to the shell." This is very true for a great number of astronomers. Emacs is much more powerful than vi, especially if you are a programmer. I recommend you take the time to work through the on-line tutorial when you have the chance. We won't take any class time on it. Here's a quick summary:

XEmacs is the variant of Emacs, the GNU super editor, included with Caldera OpenLinux. XEmacs provides additional enhancements when used under a graphical environment, such as X (KDE),



hence its "X" denomination. Under a terminal, it is almost exactly the same as regular Emacs. Any reference to Emacs applies in terms of usage to XEmacs.

Emacs is the most powerful of editors. It is available for UNIX, Windows, and a slew of other operating systems. It provides features too numerous to cover in 500 pages just dedicated to Emacs alone!

Emacs is an extensible editor. Using a programming language called Lisp, Emacs can be easily extended to do just about anything thinkable.

Many hardcore UNIX people use only one program: Emacs. They start it when they start their work session, and they close it when they exit their system. All their work, with rare exceptions, is handled through Emacs, including access to the shell.

From Emacs you can read, write, and organize your email and calendar, manipulate and organize your files, browse the Web, read news, develop software, transfer FTP files across the network, and much more. No matter what you do under UNIX, it is likely that you can do it through Emacs. Of course, Emacs also edits files. If you get the sense that Emacs is really an environment, you are correct. That is what separates Emacs from other editors.

Emacs does everything through modes. A *mode* is just a specialized interface for interacting with a specific program or for working with a specific file format such as text or programming source code. Modes are tailored to the needs of a particular task. It may sound complicated, but it really isn't. The modes are automatically handled by Emacs depending on what you are doing and the file you are editing.

Starting XEmacs and the XEmacs Tutorial

To enter Emacs (XEmacs), just type **xemacs** at the command prompt. Emacs greets you with a welcome message and a list of options. Most keystrokes under Emacs work as follows:

Ctrl+letter letter

When you see a command listed as **C-t h**, it means that first you must press **Ctrl+t** and then **h** by itself. Yes, Emacs has a lot of command keys, and it requires a little time to get used to it; however, that is time well spent.

To exit Emacs, press **Ctrl+c Ctrl+x**; this returns you to your shell. Teaching Emacs is beyond the scope of this book, but suffice it to say that it is worth exploring if you are going to do a lot of text editing.

One impressive feature of Emacs is that you can edit files through FTP as if they were local on your disk. For users working on Web-related services, this feature rocks. If your files can be reached through FTP, they can be edited and manipulated just as if they were local files. However, that is an advanced topic for another book.

If you are a programmer, Emacs does source formatting on-the-fly, so your source code is always formatted in a way that you can read. When reading other people's code, the ability to reformat the code in a way readable to you is invaluable. Emacs also provides an IDE (Integrated Development Environment): You can build your program and debug it right from Emacs.

Emacs also has an online tutorial. To access the tutorial, start Emacs, and in the welcome window type **C-h t** (press **Ctrl+h**, and then press **t**).

During Friday's long class period, we will be working with more complicated features of Linux (after a bit of review) and getting our directories and control files set up for running IRAF. We will also be working on your first assignment that will be turned in and graded that will test your proficiency with some of the fundamentals of Linux.

Linux Quick Reference

SOME USEFUL COMMANDS

Command	Task
---------	------

File/Directory Basics

ls	List files
cp	Copy files
mv	Rename files
rm	Delete files
ln	Link files
cd	Change directory
pwd	Print current directory name
mkdir	Create directory
rmdir	Delete directory

File Viewing

cat	View files
less	Page through files
head	View file beginning
tail	View file ending
nl	Number lines
od	View binary data
xxd	View binary data
gv	View Postscript/PDF files
xdvi	View TeX DVI files

File Creation and Editing

emacs	Text editor
vim	Text editor
umask	Set default file protections
soffice	Edit Word/Excel/PowerPoint docs
abiword	Edit Word documents
gnnumeric	Edit Excel documents

File Properties

stat	Display file attributes
wc	Count bytes/words/lines
du	Measure disk usage
file	Identify file types
touch	Change file timestamps
chown	Change file owner
chgrp	Change file group
chmod	Change file protections
chattr	Change advanced file attributes
lsattr	List advanced file attributes

Command	Task
---------	------

File Location

find	Locate files
slocate	Locate files via index
which	Locate commands
whereis	Locate standard files

File Text Manipulation

grep	Search text for matching lines
cut	Extract columns
paste	Append columns
tr	Translate characters
sort	Sort lines
uniq	Locate identical lines
tee	Copy stdin to a file and to stdout simultaneously

File Compression

gzip	Compress files (GNU Zip)
compress	Compress files (Unix)
bzip2	Compress files (BZip2)
zip	Compress files (Windows Zip)

File Comparison

diff	Compare files line by line
comm	Compare sorted files
cmp	Compare files byte by byte
md5sum	Compute checksums

Disks and Filesystems

df	Show free disk space
mount	Make a disk accessible
fsck	Check a disk for errors
sync	Flush disk caches

Backups and Remote Storage

mt	Control a tape drive
dump	Back up a disk
restore	Restore a dump
tar	Read/write tape archives
cdrecord	Burn a CD
rsync	Mirror a set of files

Linux Quick Reference

SOME USEFUL COMMANDS

Command	Task
---------	------

Printing

lpr	Print files
lpq	View print queue
lprm	Remove print jobs

Spelling Operations

look	Look up spelling
aspell	Check spelling interactively
spell	Check spelling in batch

Processes

ps	List all processes
w	List users' processes
uptime	View the system load
top	Monitor processes
xload	Monitor system load
free	Display free memory
kill	Terminate processes
nice	Set process priorities
renice	Change process priorities

Scheduling Jobs

sleep	Wait for some time
watch	Run programs at set intervals
at	Schedule a job
crontab	Schedule repeated jobs

Hosts

uname	Print system information
hostname	Print the system's hostname
ifconfig	Set/display network information
host	Look up DNS
whois	Look up domain registrants
ping	Check if host is reachable
traceroute	View network path to a host

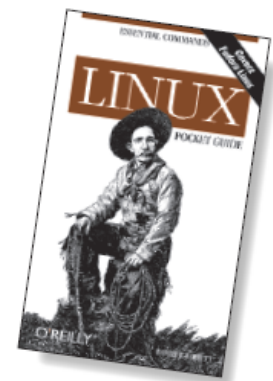
Command	Task
---------	------

Networking

ssh	Securely log into remote hosts
telnet	Log into remote hosts
scp	Securely copy files between hosts
sftp	Securely copy files between hosts
ftp	Copy files between hosts
evolution	GUI email client
mutt	Text-based email client
mail	Minimal email client
mozilla	Web browser
lynx	Text-only web browser
wget	Retrieve web pages to disk
slrn	Read Usenet news
gaim	Instant messaging/IRC
talk	Linux/Unix chat
write	Send messages to a terminal
mesg	Prohibit talk/write

Audio and Video

grip	Play CDs and rip MP3s
xmms	Play audio files
cdparanoia	Rip audio
audacity	Edit audio
xcdroast	Burn CDs



O'REILLY
www.oreilly.com

Excerpted from Linux Pocket Reference