

# An Introduction to UNIX/LINUX

Gary Lackmann, updated *Fall 2010*

The purpose of this document is to provide meteorology students with a brief introduction to the UNIX operating system. UNIX (or similar Linux) is the operating system of choice for a growing number of scientific applications, especially in meteorology. Therefore familiarity with UNIX will be advantageous throughout your career as a meteorologist, and you would do well to develop the skills needed to list it on your resume. First, we will discuss the UNIX environment, and provide a brief history of its development. Then we will review some basic aspects of our Departmental “Weather Observatory”, including a discussion of editors and a summary of some often-used UNIX commands. *The discussion will assume that you have not used any UNIX system previously, but I will assume that you have had at least some exposure to computers.* It may help to sit at a workstation as you read section 3, typing the various commands as you go. Note that most of the commands listed here work for both UNIX and Linux; the new PC workstations in the Weather Observatory are all running a version of Red Hat Linux.

## 1. The UNIX Operating System

UNIX is an *operating system*, like Windows (for PCs), VMS (for VAX systems), etc. An operating system allows users to issue commands to a computer without having to deal with the lowest-level machine language that the computer hardware actually uses. Most operating systems are written in an assembly language that is specific to the hardware platform being used. However UNIX is largely machine independent, a major advantage. Another advantage that we will discuss below is that UNIX allows easy customization of the user environment to suit individual tastes.

UNIX was originally developed by AT&T Bell Laboratories in the late 1960s, and other versions have since been developed by computer scientists at UC Berkeley. There are several versions of UNIX, including UNICOS (Cray), AIX (IBM), and ULTRIX (DEC), and, recently, LINUX (for the PC platform). Therefore, you may encounter slight differences between UNIX on different types of machines. However most of the basic commands will be the same.

## 2. The UNIX Environment

There are several “layers” in the UNIX environment, from the “lowest” (farthest removed from the user), which is called the kernel, to the “highest”, the graphical user interface (GUI). Each of these layers will be described very briefly below.

### 2.1 The Kernel

The kernel is the lowest layer of the operating system, and accounts for hardware devices, data storage, and executing other regularly scheduled tasks. This layer is machine *dependent*, unlike the rest of the operating system. Individual tasks that a user

performs constitute separate *processes*. Users usually run many processes concurrently during normal operations.

## 2.2 The Shell

Users of UNIX systems do not interact directly with the kernel. User commands are sent to the kernel via a *shell*. This part of the operating system is a high-level programming language that interprets user commands, executes the appropriate program, sends requests to the kernel, and delivers the resulting output to the user. The shell is an interface that lets you customize your user environment and automate complex operations.

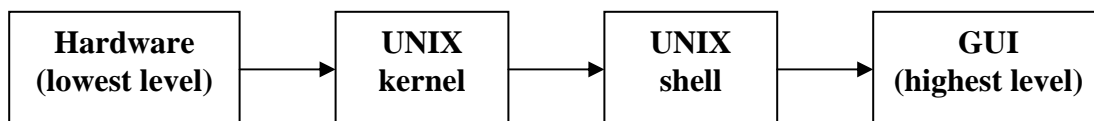
UNIX features several different types of shell that you can choose from, including the C shell (also known as the Berkeley shell), the Tahoe C shell (the T shell), the Bourne shell, and the Korn shell. I recommend using the C or T shell; these two are nearly identical except the T shell has the advantage of allowing users to edit commands with the arrow keys (e.g., you can repeat previous commands by simply hitting the up arrow). I believe that all unity accounts are set up with the T shell by default when your account is first established.

If you use either the C or T shell, you will have a file in your root directory (the directory you are automatically in when you first log in) named `.cshrc` (on unity it is actually called `.mycshrc`). This file is your *c-shell record* file. In this file, you can set up custom commands (aliases) and set up links to GEMPAK and other software programs. In a moment you will edit this file and add a few convenient aliases. Note that the period character at the start of the file name (`.cshrc`) causes this file to be “hidden” when you list all the files in a given directory unless you use additional options when listing the files.

## 2.3 The Graphical User Interface (GUI)

The GUI is the highest level of the UNIX operating system. It is simply the window display environment that you see on the screen when using a workstation. You can customize this environment to suit your tastes as well, for example, you can set the background color, font size, window size, and have a clock or calendar automatically appear when you log in. If you have multiple windows open simultaneously, you will have to use the cursor to “activate” the window you want to use. A mouse controls the movement of the cursor; I will assume that you have enough computer familiarity to master this aspect of the system!

To summarize the UNIX system environment, consider the following diagram, which demonstrates the layered structure of UNIX:



### 3. Some Basics (that you probably already know)

Log in to a workstation.

Place the cursor in a terminal window and type the command

**ls** (Hereafter, all commands will appear in bold text)

This is the UNIX command for **listing** all the files in the current directory. The first time you log in, there may not be any visible files listed. However, most UNIX commands have additional options or switches. For example, now try the command

**ls -al**

This is a version of the **ls** command that shows a more complete directory listing. The **-a** option specifies that all files will be listed, including "hidden" files (those that start with a period, such as **.cshrc** and **.login**). The **-l** option gives the long version of the listing, including the size and date associated with each file. Note that all files beginning with a **.** are "hidden" files, and will not show up with a regular **ls** command.

To obtain more information on the **ls** command (or any other UNIX command), type **man ls**, and you will be surprised at how many variations on this simple command are possible! You will quickly discover that UNIX is not a very simple operating system.

For many MEA 443 class projects, you will need to have some free disk space. To find out how you're doing with this, use the command

**quota .**

If you have too many files, you will need to **remove** some of the using the **rm** command. Use this one with care! It is a good idea to set an alias so that there is an interactive check before a file is deleted. I'll discuss this in more detail below when I talk about the **.mycshrc** file and how to use it.

Another very useful command is the **cp** command, which allows you to **copy** one file to another. Try typing

**cp .mycshrc .mycshrc\_backup**

This will copy your **.mycshrc** file, which you will modify to customize your environment, to a new file called **.mycshrc\_backup**. If you botch the next part of the exercise you will have a backup of your original **.mycshrc** file! Now, type **ls -al** to see if the new file is listed.

### 4. Editors

One of the first things you should learn how to do is use one of the system editors. Our local system has several editors, but the one I use is called **emacs**. Another standard editor used by many UNIX users is called **vi**. Many students prefer **nedit** (**note: nedit seems to no longer be available... use emacs instead**). If you are interested in learning more about either vi or emacs, type **man vi** or **man emacs**. To use emacs, type

**emacs filename**

where filename is the name of a new or pre-existing file that you want to modify. For example, type

**emacs test.file**

This will spawn a new window containing the contents of the file being edited, in this case a file called "test.file". In order to modify the file, make sure that the cursor is in the new emacs window (you may have to click the left mouse button to "activate" the window). You can use the arrow keys to move around, etc. Type in a few lines of text to test out the editor; when you have finished, type the emacs command **<ctrl> x**, **<ctrl> c**, and answer **y** (for yes) if you want to save the changes. [that is, hold down the **<ctrl>** (control) key while pressing x, then do it again while pressing c]. **Note: you can also use the pull-down menus to accomplish these things if you don't like to memorize key strokes.** Other useful emacs commands include **<ctrl> s** (searches for a string), **<ctrl> k** (deletes a line), and **<ctrl> y** (replaces last deleted line or lines). Try playing around, and be sure to try out the help and other menus at the top of the emacs window. Note that you can use the pull-down menus instead of the key commands if you are more comfortable with that.

## 5. Aliases

You are now ready to use a UNIX command called **alias**. In UNIX, you can define commands to be anything you like. For example, you could define a command **dir** to be set to really mean **ls -al** if it's easier for you to remember to type dir for a directory listing. To set up this alias, type

**alias dir 'ls -al'**

Try typing **dir** now to see if it works. If you place aliases such as this in your **.cshrc** file, they will automatically be defined when you log in, and you can customize your command set any way you like. Let's start by making an alias for netscape, the web browser you will be using most frequently. To do this, edit your **.mycshrc** file by typing

**emacs .mycshrc**

An emacs window will appear with the contents of your **.mycshrc** file. Move the cursor down a few lines, and insert a line that begins with a **#**. The **#** is a comment signal, which

tells the computer to ignore the rest of that line. This is convenient for writing comments or notes to your self within UNIX files. Type

### **# an attempt at using aliases**

This is just a comment to remind you that the following line will contain an “alias”, or user-defined command. Then, go to the end of that line and hit enter, creating a blank line. In the blank line, type

```
alias mz 'mozilla &'
```

This sets up an alias (called mz in this case) that is really the command “mozilla &”. Note that you can replace the second word (mz) with anything you want. Then, the next time you log in, you could just type “mz” instead of “mozilla &”. In the above example, the & makes mozilla run in “background mode”, meaning that you can use the window where you run mozilla for other things as well.

Now, exit emacs by typing **<ctrl>x <ctrl>c y**, (or, use the pull-down menu commands) and then to activate your modified .cshrc file, type

```
source .mycshrc
```

Note that in general, after modifying your .cshrc file, you must always type **source .mycshrc** to activate the changes. Now, try typing **mz**

Be patient, sometimes it takes a minute for netscape to pop up. Note: if you run netscape and GARP or NMAP at the same time, the two different software programs may compete for the same color table, creating a conflict and causing images to appear blank in GARP. This should be much less of a problem with the new workstations, compared to the old Sun workstations we had in past years.

## **6. Some Basic Commands and Man Pages**

In UNIX, each command can have up to several *arguments* that follow it. These allow different options or switches associated with the command to be invoked. For example, as we discussed above, the command to list a directory of files is **ls**. You can use arguments to customize the ls command to perform the way you like. For example, you could type **ls -al** for a more complete listing of files. With any command, you can obtain a complete list of arguments, related commands, and examples of usage by typing **man command-name**. “Man” is short for **man**ual; the UNIX man pages are VERY detailed, in fact they are too detailed in my opinion, but they are complete. If you do not know a command, but want to find out how to do something, type

```
man -k topic
```

where *topic* is the subject that you wish to learn more about. The **-k** option specifies that the following input will be a “keyword” that specifies the topic.

Here is a list of some of the basic UNIX commands that you will most likely use:

command	function	usage
man	UNIX help facility	man -k <i>topic</i> or man <i>command</i>
ls	directory listing	
mv	renames (moves) file	mv oldfile.txt newfile.txt
cp	copies files	cp origfile.txt copyfile.txt
rm	deletes (removes) file	rm file.txt
cd	change directory	cd newdirectory
mkdir	create new directory	mkdir mea443
rmdir	removes directory	rmdir mea443
df -k	check disk status	
du -sk *	show your disk usage, including subdirectories	
/bin/ps	list your processes	
kill	stops processes	kill -9 <i>process_id</i>
cat	list contents of file	cat file1 >> file2
more	view contents of file	more filename.txt
head	view first 10 lines of file	head filename.txt
tail	view last 10 lines of file	tail filename.txt
pg	displays file page at a time	
diff	shows differences in files	diff file1 file2
file	displays type of file	file filename
chmod	changes file mode	chmod u+x file
> <	in general, chmod (ugo) +/- (rwx) see man pages	more file1.txt > file2.txt
grep	directed output or input	grep <i>string</i> filenames
emacs	search for string in file	emacs filename
vi	editor program	vi filename
ssh	another editor	ssh <i>remote_ip_number or machine</i>
sftp	login to remote host	
awk	transfer files to/from remote host	
cut, paste, sort	an extremely powerful file/data manipulation program	
who	other file manipulation programs	
pwd	show current users	
setenv	shows current directory (present working directory)	setenv DISPLAY sun2:0
history	set environmental variable	check man setenv for other options
	see list of past commands	

## 7. UNIX Scripts

If there is a series of UNIX commands that you run on a routine basis, you can combine them into a file known as a *script*. For example, you could edit a file called test.script and put the following lines in it:

```
ls -al > dir.txt
elm userid < dir.txt
exit
```

and you would have a script that 1) wrote the contents of your current directory into a file called dir.txt, then 2) emailed this file to the user *userid* using the elm mailer. This is intended only as an example, and is not a very useful thing to do, however, there are many other situations where scripts are extremely useful, especially those involving GEMPAK plots. We will be using UNIX scripts to execute a sequence of GEMPAK commands on a regular basis later on in the semester.

## 8. Usage of the Weather Observatory Computers

There are a few “rules” which we ask you to respect regarding the usage of the lab computers. First, if one of the computers freezes up, don't just turn it off. There are gentler ways to re-boot a workstation, and simply turning it off could damage the system.

Second, do not play tricks on your fellow users. There are funny things that you can do to your pals, but the prank may not be funny to your friend if it involves the accidental destruction of their class project, for example!

Inappropriate uses of the network include:

- attempting to interfere with the performance of the system
- lending your computer account to another person
- using the account of another person
- attempting to access another person's files without their permission
- interfering with the work of another user
- attempting to circumvent system security

Remember that you are personally responsible for any activity on the network corresponding to your *userid*; therefore it is important to ***keep your password private*** at all times! Change your password often, and choose passwords that are easy to remember but difficult for someone else to guess. It is a good idea to include some special characters in your password.