

A Short Introduction to Unix

Adapted from:

www.med.nyu.edu/rcr/rcr/course/PPT/unix_lec1.ppt

Computers are powerful tools for geophysics research

- Ever increasing reliance on increased computing power for geophysical studies
- As geophysical data sets grow larger and problems become more complex, computing power has also grown (not independently).
- Because of its efficiency, the most powerful computers run Unix
 - Linux, is a Unix-based operating system that is rapidly replacing traditional UNIX because of lower cost, more rapid evolution, and ease of use

Unix Advantages

- It is very popular, so it is easy to find information and get help
 - Books available at local bookstore/internet
 - Numerous websites
 - USENET discussions and e-mail lists
 - Most Comp. Sci. students know Unix
- Unix can run on virtually any computer
(Sun, SGI, x86, Macintosh, etc)
- Unix is free or nearly free
 - Linux/open source software movement, numerous versions of the OS and most all tools are free, or have a free version
 - Linux has caused traditional Unix companies to reduce costs dramatically (e.g., Sun, SGI)

Stable and Efficient

- Unix is very stable:
 - Crashes are very rare
 - Software failures generally do not require reboot
- Unix is very efficient
 - gets maximum number crunching power from the processor(s)
 - it can smoothly manage extremely huge amounts of data
 - Will run well on old systems that get bogged down with modern Windows releases
- Most geophysical tools are developed to run on Unix and Linux machines.
- Matlab programs are an exception because it runs on either Unix or Windows.

Unix has some Drawbacks

- Unix computers are controlled by a command line interface
 - Steeper learning curve than windows or Mac
 - + This is changing, on Linux systems, most tools have GUI interfaces that make it easier
 - + from command line, operations use less resources and are easier to perform/manage remotely
- Numerous versions of Unix with subtle (or not so subtle) differences
 - This is particularly the case between Unix and Linux
 - Linux is less standardized but evolving much more rapidly (more user-friendly and more powerful)

General Unix Tips

- UNIX is case sensitive!!
 - **myfile.txt** and **MyFile.txt** do **not** mean the same thing
 - Default directory listings do not separate files from directories like Windows. They list all structures in alphabetical order starting with numbers
- Every program is independent
 - the core operating system (known as the *kernel*) manages each program as a distinct process with its own little chunk of dedicated memory.
 - If one program runs into trouble, it dies, but does not affect the kernel or the other programs running on the computer.

The Unix Shell

- This is the traditional way to communicate with a Unix computer, through a command program known as a *shell*. Similar to Windows *cmd* or dos prompt.
- The shell interprets the commands that you type on the keyboard.
- Numerous types of shells to work from(e. g., sh, bash, csh, tcsh)
- You can use shell scripts to write simple programs to automate many tasks

Unix Commands

- Unix commands are short and cryptic like *vi* or *rm*.
 - Will take some learning.
- Every command has a host of modifiers which are generally single letters preceded by a hyphen:
ls -l or *mv -R*
 - Capital letters have different functions than small letters, often completely unrelated.
 - A command also generally requires an argument, meaning some file on which it will act:
cat -n filename.txt

Wildcards

- You can substitute the * as a wildcard symbol for any number of characters in any filename.
- If you type just * after a command, it stands for all files in the current directory:

*lpr ** will print all files

- You can mix the * with other characters to form a search pattern:

ls a.txt* will list all files that start with “a” and end in “.txt”

- The “?” wildcard stands for any single character:

cp draft?.doc will copy *draft1.doc*, *draft2.doc*, *draftb.doc*, etc.

Typing Mistakes

- Unix is remarkably unforgiving of typing mistakes
 - You can do a lot with just a few keystrokes, but it can be hard or impossible to undo
- If you have not yet hit ‘return’
 - The ‘delete’ key removes the characters that you just typed
 - Generally you can use the arrow keys to move back and forth on the command line to correct a mistake without deleting the rest

Getting Help in Unix

- Unix is somewhat user-friendly
 - While not actively user-hostile, it is perfectly willing to sit there and taunt you with a blank screen and a blinking `>` cursor.
- There is a rudimentary Help system which consists of a set of "manual" pages for every Unix command.
- The *man* pages tell you which options a particular command can take, and how each option modifies the behavior of the command.
- Type *man* and the name of a command to read the manual page for that command.

```
anewman@terremoto /home/anewman - Shell - Konsole <3>
Session Edit View Bookmarks Settings Help
man(1) Manual pager utils man(1)

NAME
man - an interface to the on-line reference manuals

SYNOPSIS
man [-c|-w|-tZHT device] [-adhu7U] [-il-I] [-m system[,...]] [-L
locale] [-p string] [-M path] [-P pager] [-r prompt] [-S list] [-e
extension] [[section] page ...] ...
man -l [-7] [-tZHT device] [-p string] [-P pager] [-r prompt] file ...
man -k [apropos options] regexp ...
man -f [whatis options] page ...

DESCRIPTION
man is the system's manual pager. Each page argument given to man is
normally the name of a program, utility or function. The manual page
associated with each of these arguments is then found and displayed. A
section, if provided, will direct man to look only in that section of
the manual. The default action is to search in all of the available
sections, following a pre-defined order and to show only the first page
found, even if page exists in several sections.

The table below shows the section numbers of the manual followed by the
types of pages they contain.

1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions eg /etc/passwd
6 Games
7 Miscellaneous (including macro packages and conven
tions), e.g. man(7), groff(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]

A manual page consists of several parts.

They may be labelled NAME, SYNOPSIS, DESCRIPTION, OPTIONS, FILES,
SEE ALSO, BUGS, and AUTHOR.

less: (stdin) -- lines 1 - 44
```

More Help (?)

- The man pages give information about specific commands
- So what if you don't know what command you need?
- *apropos* will give you a list of commands that contain a given keyword in their man page header:
 - *apropos password*
 - The *man* command with the *-k* modifier gives a similar result to *apropos*
- Look at getting a Unix book
 - O'Reilly books are usually good (e.g. “Unix in a nutshell”)
- Google for solutions

Unix Help on the Web

Here is a list of a few online Unix tutorials:

- Unix for Beginners

<http://www.ee.surrey.ac.uk/Teaching/Unix/>

- Getting Started in Unix

<http://www2.ucsc.edu/cats/sc/help/unix/start.shtml>

- Unix Guru Universe

<http://www.ugu.com/sui/ugu/show?help.beginners>

- Getting Started With The Unix Operating System

<http://www.leeds.ac.uk/iss/documentation/beg/beg8/beg8.html>

Logging in to the Terremoto (Andy's Linux Box)

From an Ethernet connection within EAS (not the same as Wireless connection)

1. start 'Xwin32' (should be installed on Windows). As a default, this runs an X-server on your machine, allowing remote Unix windows to be displayed
2. Use SecureCRT (should be installed on Windows)
 - Create a new session (**SSH2**)
 - Hostname: **terremoto.eas.gatech.edu**
 - Port:22, Firewall: **None**
 - Username:*I will give this to you (e.g. anewman)*
 - Session name: **username@terremoto** is a good choice
 - When installed click on properties for that session, go to 'port forwarding->Remote/X11' and click 'Forward X11 packets'
 - Now start session **username@terremoto**
 - If asked to create a public key say **no**
 - You will be prompted for your password (we will set this up together)
 - **You should now be connected**
- Test your connection.
 - Type **xterm** at the shell prompt and see if another shell screen comes up. If not, check that you have allowed X11 packets to be forwarded and that Xwin32 is running.

Filename Extensions

- Most UNIX filenames start with a lower case letter and end with a dot followed by one, two, or three letters: *myfile.txt*
 - However, this is just a common convention and is not required.
 - It is also possible to have additional dots in the filename.
- The part of the name following the dot is called the “extension.”
- The extension is often used to designate the type of file.

Some Common Extensions

- By convention:
 - files that end in *.txt* are text files
 - files that end in *.c* are source code in the "C" language
 - files that end in *.html* are HTML files for the Web
 - Compressed files have the *.zip* or *.gz* extension
- Unix does not require these extensions (unlike Windows), but it is a sensible idea and one that you should follow

Working with Directories

- Directories are a means of organizing your files on a Unix computer.
 - They are equivalent to folders in Windows and Macintosh computers
- Directories contain files, executable programs, and sub-directories
- Understanding how to use directories is crucial to manipulating your files on Unix.

Your Home Directory

- When you login to a Unix system, you always start in your Home directory.
- Create sub-directories (using **mkdir**) to store specific projects or groups of information, just as you would place folders in a filing cabinet.
- Do **not** accumulate thousands of files with cryptic names in your Home directory

File & Directory Commands

- This is a minimal list of Unix commands that you must know for file management:

ls (list)

mkdir (make directory)

cd (change directory)

rmdir (remove directory)

cp (copy)

pwd (present working directory)

mv (move, rename)

more, less (view by page)

rm (remove)

cat (view entire file on screen)

- All of these commands can be modified with many options. Learn to use Unix '*man*' pages for more information.

Navigation

- *pwd* (**present working directory**) shows the name and location of the directory where you are currently

working:

```
> pwd  
/home/aneuman
```

- This is a “pathname,” the slashes indicate sub-directories
- The initial slash is the “root” of the whole filesystem

- *ls* (**list**) gives you a list of the files in the current directory:

```
> ls  
Misc      test2.txt  
bin       temp      testfile
```

- Use the *ls -l* (**long**) option to get more information about each file

```
> ls -l  
total 1768  
drwxr-x--- 2 browns02 users 8192 Aug 28 18:26 Opioid  
-rw-r----- 1 browns02 users 6205 May 30 2000 af124329.gb_in2  
-rw-r----- 1 browns02 users 131944 May 31 2000 af151074.fasta
```

Sub-directories

- ***cd*** (**change directory**) moves you to another directory

```
>cd bin
> pwd
/home/aneuman/bin
```

- ***mkdir*** (**make directory**) creates a new sub-directory inside of the current directory

```
> ls
assembler phrap  space
> mkdir subdir
> ls
assembler phrap  space  subdir
```

- ***rmdir*** (**remove directory**) deletes a sub-directory, but the sub-directory must be empty

```
> rmdir subdir
> ls
assembler phrap  space
```

Shortcuts

- There are some important shortcuts in Unix for specifying directories
 - `.` (dot) means "the current directory"
 - `..` means "the parent directory" - the directory one level above the current directory, so *cd* `..` will move you up one level
 - `~` (tilde) means your Home directory, so *cd* `~` will move you back to your Home.
 - Just typing a plain *cd* will also bring you back to your home directory

Unix File Protections

- File protection (also known as permissions) enables the user to set up a file so that only specific people can read (**r**), write/delete (**w**), and execute (**x**) it.
- Write and delete privilege are the same on a Unix system since write privilege allows someone to overwrite a file with a different one.

File Owners and Groups

- Unix file permissions are defined according to ownership. The person who creates a file is its owner.
 - You are the owner of files in your Home directory and all its sub-directories
- In addition, there is a concept known as a Group.
 - Members of a group have privileges to see each other's files.
 - This can be done for people working on a joint project or to otherwise share data/programs

View File Permissions

- Use the *ls -l* command to see the permissions for all files in a directory:

```
> ls -l
drwxr-x--- 2 browns02 users 8192 Aug 28 18:26 Opioid
-rw-r----- 1 browns02 users 6205 May 30 2000 af124329.gb_in2
-rw-r----- 1 browns02 users 131944 May 31 2000 af151074.fasta
```

- The username of the owner is shown in the third column. (The owner of the files listed above is **browns02**)
 - The owner belongs to the group “**users**”
- The access rights for these files is shown in the first column. This column consists of 10 characters known as the attributes of the file: **r**, **w**, **x**, and **-**
 - r** indicates read permission
 - w** indicates write (and delete) permission
 - x** indicates execute (run) permission
 - indicates no permission for that operation

```
> ls -l
```

```
drwxr-x--- 2 browns02 users 8192 Aug 28 18:26 Opioid  
-rw-r----- 1 browns02 users 6205 May 30 2000 af124329.gb_in2  
-rw-r----- 1 browns02 users 131944 May 31 2000 af151074.fasta
```

- The first character in the attribute string indicates if a file is a directory (**d**) or a regular file (**-**).
- The next 3 characters (**rwX**) give the file permissions for the owner of the file.
- The middle 3 characters give the permissions for other members of the owner's group.
- The last 3 characters give the permissions for everyone else (others)
- The default protections assigned to new files on our system is: **-rw-r-----** (owner=read and write, group =read, others=nothing)

Change Protections

- Only the owner of a file can change its protections
- To change the protections on a file use the *chmod* (change mode) command.

[Beware, this is a confusing command.]

- First you have to decide for whom you will change the access permissions:
 - » the file owner (**u**)
 - » the members of your group (**g**)
 - » others (**o**) (ie. anyone with an RCR account)
- Next you have to decide if you are adding (+), removing (-), or setting (=) permissions.
- Taken all together, it looks like this:

```
> chmod u=rwx g+r o-x myfile.txt
```

This will set the owner to have read, write, and execute permission; add the permission for the group to read; and remove the permission for others to execute the file named **myfile.txt**.

Commands for Files

- Files are used to store information, for example, data or the results of some analysis.
 - You will mostly deal with text files
 - Files on the RCR Alpha are automatically backed up to tape every night.
- *cat* dumps the entire contents of a file onto the screen.
 - For a long file this can be annoying, but it can also be helpful if you want to copy and paste (use the buffer of your telnet program)

more

- Use the command *more* to view at the contents of a file one screen at a time:

```
> more t27054_cel.pep
!!AA_SEQUENCE 1.0
P1;T27054 - hypothetical protein Y49E10.20 - Caenorhabditis elegans
submitted to the EMBL Data Library, August 1997
27054 Length: 534 May 30, 2000 13:49 Type: P Check: 1278 ..
1 MLKKAPCLFG SAILGLLLA AAGVLLLIGI PIDRIVNRQV IDQDFLG YTR
51 DENGTEVPNA MTKSWLKPLY AMQLNIWMFN VTNVDGILKR HEKPNLHEIG
101 PFVFDEVQEK VYHRFADNDT RVFYKNQKLY HFNKNASCPT CHLDMKVTIP
t27054_cel.pep (87%)
```

- Hit the spacebar to page down through the file
 - At the bottom of the screen, *more* shows how much of the file has been displayed
- More sophisticated options for viewing and editing text files are available in a text editor.

Copy & Move

- *cp* lets you copy a file from any directory to any other directory, or create a copy of a file with a new name in one directory
 - `cp filename.ext newfilename.ext`
 - `cp filename.ext subdir/newname.ext`
 - `cp /u/jdoe01/filename.ext ./subdir/newfilename.ext`
- *mv* allows you to move files to other directories, but it is also used to rename files.
 - Filename and directory syntax for *mv* is exactly the same as for the *cp* command.
 - `mv filename.ext subdir/newfilename.ext`
 - **NOTE:** When you use *mv* to move a file into another directory, the current file is deleted.

Delete

- Use the command *rm* (remove) to delete files
- **There is no way to undo this command!!!**
 - We have set the RCR Alpha to ask if you really want to remove each file before it is deleted.
 - You must answer “Y” or else the file is not deleted.

```
> ls
af151074.gb_pr5 test.seq
> rm test.seq
rm: remove test.seq? y
> ls
af151074.gb_pr5
```