**Geol 335.3**

# Lab #8: Introduction to UNIX and GMT

In this lab, you'll familiarize yourself with some of the leading components of scientific computing: **UNIX** operating system, and a free, open-source, GIS/plotting package called **Generic Mapping Tools** (GMT).

## *UNIX*

Program environment providing the interfaces between user applications and components of the computer (keyboard, disks, displays, network cards), etc., is called the "operating system", or OS. As you probably know, Windows is not the best OS for scientific (as well as nearly any other) computing. UNIX is the most successful and widespread OS today, characterized by its ability to operate on a vast variety of computer architectures, stability, polished networking and multiprocessor capabilities. Most "serious" geoscience applications are based on UNIX.

The implementation on an IBM-compatible PC is called Linux. In the following exercises, you will utilize one of the three Linux workstations in our lab. Note that there exist several free Linux distributions (Ubuntu, Fedora, OpenSUSE) which can be installed on any computer notebook, together with many hundreds of great software packages.

Although several graphical user interfaces are available under Linux, the essentials of UNIX could be better understood through the use of one of its command languages, called "shells".

(1) On your Linux desktop, go to "Applications" and open a "terminal" window. This is your command shell.

(2) During you first session only, you need to copy the standard student's configuration used in this lab. Do this by typing:

```
cp ../studentfiles/.cshrc .
```

(3) Type:

```
csh | tee –a csh_log.txt
```

This will start the so-called 'C shell' and duplicate your dialog with it into file *csh_log.txt*

Almost any command in UNIX is a program, and all programs operate between the input and output "streams": *input_stream* -> *program* -> *ouput_stream*. These streams can be a lot of things: data or text files, keyboard outputs, displays, tape or sound devices, network connections, etc. Your terminal window is also a stream, called "standard output", or *stdout*. Try the following commands (**note that all names in UNIX are *case-sensitive***):

(4) `pwd` (This prints into *stdout* the current directory. Note the format of the path name, with slashes `'/'` separating the subdirectories.).

(5) `mkdir lab8` (This creates a new directory `lab8`). Try typing the same again. What happens?

For the GMT exercise at the end of this lab, we have prepared several examples from actual 3D seismic data processing by graduate students. To obtain the examples, type:

```
wget http://seisweb.usask.ca/classes/GMTexamples.tgz
```

this will copy the archive of our examples into your current directory. Note that '*wget*' allows copying files or directories located anywhere on the web. Unpack the archive file by:

```
tar xvfz GMTexamples.tgz
```

this will create subdirectory *GMTexamples*. Now you can delete the archive:

```
rm GMTexamples.tgz
```

(6) List the current directory by typing '`ls`'. You will see two names: *lab8* and *GMTexamples*.

Most commands accept options, sometimes called "switches", which are usually represented by parameter strings starting with a '-' or '--' following the command name. Try:

(7) `ls –l`

This is the 'long' format showing with ownership/access attributes, creation dates, and file sizes. Note that the first 'd' in each line means this file is a directory.

(8) `ls –a -l ~`

This also shows the 'hidden' files, whose names start from a dot. Tilda '~' denotes your 'home' directory into which you always log in when starting the session. Note the various configuration files used by various programs.

To find out about the available options and any other info about the commands, use '`man`', followed by the name of the command:

(9) `man ls`

(10) `cd lab8; pwd` (This changes the working directory into ‘lab8’ and then prints the new path.).

(11) `cp ../GMTexamples/* .`

This copies all files from your *GMTexamples* directory, but not its subdirectory, into your current directory. Note that the current directory is denoted `'.'`, and the one immediately above it is `'..'`.

The input and output streams of any command can be *redirected* into files. Redirection is performed using ‘>’ (send to file) ‘<’ read from file, or `'|'` (“pipe”, sending output of one program into another). Try:

(12) `man ls > man.ls.txt`

This will send the outputs of ‘`man ls`’ into a file. To view this file, use ‘`more`’:

(13) `more man.ls.txt`

UNIX includes many useful programs for editing and formatting text files. In particular, program ‘`wc`’ counts lines, words, and symbols in its input, and ‘`grep`’ lists all occurrences of a particular pattern in its input. Try this:

(14) `grep character man.ls.txt`

or

(15) `man ls | grep character`

Both outputs should be the same, showing all the occurrences of word ‘character’ in the manual page for `ls`. Note that in the second case, we create no file for the complete listing.

The above are samples of just a few useful commands in UNIX. Now, with the use of piping and `man` command, compose command lines to perform the following tasks:

(16) (10%) List the contents of your directory `../GMTexamples/data` sorting the files by their sizes. Send the output into file `ls1.txt`

(17) (10%) Sort the values in file `../GMTexamples/data/hist.txt` and send the output into a new file `hist_sort1.txt`. This can be done by a program named `sort`. Repeat sorting in reverse numeric order and write the output into `hist_sort2.txt`

(18) (10%) Count the number of standard programs installed on your Linux system. These programs are files in directories /bin (general utilities), /sbin (system utilities), and various /*/bin (custom programs). Hint: list the contents of these directories in a single *ls* command, output one file name per line, and count the lines.

(19) (10%) Count the number of lines in the manual for program '`grep`' in which pattern 'the' occurs but pattern 'and' does not. Hint option '-v' reverses the sense of matching in *grep*.

## *GMT*

Generic Mapping Tools (GMT) is a popular among geoscientists UNIX package that creates maps and quality plots. It operates by executing a series of commands in the spirit of those described above, each adding a layer to the output plot. The plots use PostScript format that can be viewed using many viewer programs (*e.g.*, `ghostview`) and printed using `lpr –h` (or keys `Ctrl-P` in most interactive programs). They can also be input into Corel DRAW and similar programs, and edited.

(17) Look up 'Generic Mapping Tools' in *Google* and have a look at its documentation pages, and examples.

Since we have no time to study GMT in any detail, we'll just run a couple examples to get a feeling for it.

All programs in GMT are self-documented. Each program responds with its description when invoked without parameters. Type:

(18) `GMT`

This gives a general description of the package. To display it in an orderly, page-by-page fashion, use `GMT | more`.

(19) `pscoast | more`

This program plots the coastlines, rivers, and state boundaries for a selected region). Note the numerous switches used to control the region, map scales, and projections.

(20) (10%) Execute the following example (you can cut and paste it from here by using the left and middle mouse buttons):

```
gmtset BASEMAP_TYPE FANCY DEGREE_FORMAT 3 GRID_CROSS_SIZE 0.05
pscoast –R-130/-70/24/52 –Jl-100/35/33/45/1:50000000 –B10g5 –Dl –G200 \
       –A500 –N2/1 –P -X0.5 –Y0.5 > usa.ps
```

*(Note that the backslash '\' at the end of the second line, followed by hard return, simply means that this line is continued in the following line. You can type the entire call to* `pscoast` *in one line).*

This should result in a map of the USA in Lambert's Conic Conformal projection. View this map by using:

```
ghostview usa.ps
```

or

```
evince usa.ps
```

Note that it would be a good idea to first type all of the above into a file, *e.g.*, '`example.gmt`', and then execute the file by using the UNIX C-shell command *source*:

```
source example.gmt
```

(21) (20%) Modify the script above to show more of Canada and to send the output into file `canada.ps`. You will have to look into the meanings of options –R and –J. To find out about these meanings, type, for example, `pscoast` in the console window.

There are several text editors available in Linux, such as *gedit*, *kate* (these are fairly intuitive to anyone familiar with Windows' editors), *emacs*, or *vi* (for UNIX gurus, try `more vi` first). By the way, OpenOffice (go to 'Start'->Office) is a complete suite of integrated word processing and presentation applications (generally) compatible with MS Office.

Now change directory to *GMTexamples* we created at the beginning of this lab (make sure you understand this command):

```
cd ~/GMTexamples
```

(22) (5%) Look into file `geometry.sh`. This script plots source and receiver positions, and also several horizontal wells used in Weyburn $CO_2$ sequestration monitoring. Note how UNIX program '`awk`' is used to selectively extract columns of numbers from data files. This is a 'Bourne shell' script, and you need to execute it like this:

```
sh geometry.sh
```

This script produces PostScript file `geometry.ps` and displays it using the viewer program named 'evince'.

(23) (5%) Look into `color_map.sh`. This script reads in seismic amplitudes (from the reservoir caprock) from a file, performs substantial processing (smoothing, editing), and plots the result in a colour image. Try it:

```
sh color_map.sh
```

(5%) In this script, you will find a call to program `makecpt` which builds a colour palette. Try a couple of other palette names.

(24) Look into `histogram.sh`. This script illustrates a handy `pshistogram` program which allows calculating and drawing histograms. This histogram shows the distribution of amplitudes in the same data file `data/data.txt` as used in `color_map.sh`. Execute it by:

```
sh histogram.sh
```

(5%) Type `pshistogram` without parameters to see its key options. This histogram shows the distribution of amplitudes in the same data file `data/data.txt` as used in `color_map.sh`. Change the bin widths to 0.02 and plot it again.

(25) Exit the C shell by typing `exit`. You can now use `more` or other programs to view the log of your work in file *csh_log.txt*

## *Hand in:*

csh_log.txt, other text and PostScript files created above by uploading to class web page.