Linux and vi Helper Sheet

This handout is intended to give you helpful information for working in Linux and using vi.

Linux Commands (that you are likely to need to do assignments in this class)

man – manual page – This command displays the manual page for most Unix commands and the standard C library commands as well. Just type man followed by the command and it will display it. Type ‘q’ to get out of that display or use arrows, space bar, and return key to move around.

ls – list – This command does a list of the current directory. I like to use the option ‘ls –l’ which shows details of the files and which I find more readable and ‘ls –al’ which lists hidden files in the long format.

pwd – present working directory – This prints out the full path for the directory you are in.

mkdir – make directory – This command takes one argument which is the name of a directory to create. It creates that directory in the current directory.

cd – change directory – This command allows you to move between directories. You type cd followed by the directory you want to go to. Names of directories are separated by slashes, ‘/’. The directory above the current one is denoted by ‘..’.

rm – remove a file – this command removes the specified file. You can use ‘wildcards’ here. For example ‘rm s*’ removes all files that begin with s and ‘rm assn?.cpp’ removes all files where the ‘?’ is replaced by any character.

rmdir – remove directory – takes one argument which is the name of the directory to remove. It must be empty first.

mv – move – This command takes two arguments and moves the file specified by the first argument to the name or location specified by the second argument.

vi – This is the editor you will probably use for this class. When you are in the lab you can use the variant gvim or if you wish you could use gedit might be more like a normal editor for you.

more – This prints out a text file and pauses at the end of each screen. Hit return to move down one line and space to move down a full screen. Hit q to jump out at any time.

less – This is actually an advanced version of more which allows you to use arrows to move up and down. Against press q to jump out.

grep – Takes two arguments. The first is a string to search for. The second is the file or files to search in.

vi commands (that I can think of off the top of my head)

i – Insert typing before the current character.
a – Append typing after current character.
A – Append typing at end of line.
Esc – Leave an insert mode.
:w – write/save the file.
:wq – write/save the file then quit.
:q – Just quit (won’t let you do this if there are unsaved changes so you have to use :q!).
x – delete the character .
dd – delete a line and put it in the clipboard. (ndd to delete n lines)
yy or Y – Yank a line to the clipboard. (nyy to yank n lines)
p – Paste the clipboard contents after the current line.
P – Paste the clipboard contents before the current line.
cw – Change the “word” beginning with the current location (deletes up to next white space and then goes into insert mode).
u – Undo the last command.
r – Replace a single character.
R – Replace multiple characters.
/?? - Search for the text/string ??.
n – Repeat the last search.
. – Repeat the last command.
:n – Jump to line n in the file.

All of the commands without a : in front of them can be preceded by typing a number and they will be repeated that many time.

Redirecting input and output in Unix/Linux

One of the great powers of Unix is that it is very easy to take the output of one program and use it as the input for another program. It is also easy to have the input or output go from or to a file. To make the output of one program go to another you use what is called a pipe. It is represented by the character '|'. When one program gives a printed output you can send that to another by following the first command with a pipe and the second command. For example, if a directory listing is long you might consider doing this

```
ls –l | more
```

So that it pauses after each screen full. You can also redirect output to a file with the '>' character or make input come from a file with the '<' character. When we talk more about testing I will have you develop test suites for your programs. Those are series of inputs that you will want to run through them. To make this easier you can type the inputs for one test run into a file once and send that file in as the input every time. For example, if you have an executable a.out and one set of inputs is written in a file test1.txt you might run this command

```
a.out < test1.txt
```

That would feed in those inputs without to having to retype all of them. If your program prints out large amounts of information and you want to be able to look at it in vi you could do something like

```
a.out > output.txt
```

then run vi on output.txt to examine the output from the program.

Submitting homework by e-mail

For the assignments you will hand in printed copies of the analysis, design, and source code the day it is due (I should receive them that day). You will also need to e-mail me your source code. To do this you can use the mail command and the ability to redirect input. For example, to send me a program file called test.cpp you would type in

```
mail mlewis@bianca.cs.trinity.edu < test.cpp
```

Please use that account and not just mlewis@trinity.edu. It will make my life much easier.