DIRECTORY STRUCTURE IN UNIX

Recall that directories are organized as trees, with subdirectory branches and sub-branches.

Tilde '~' is your root directory; the topmost one in your directory tree.
Dot '.' is your current directory, wherever you are in your tree.
The directory right above the one you're currently in is '..'

You can specify the same directory in two ways. The first way is to give the path to the file with reference to the root directory ('~'):

```
emacs ~/.cshrc &            ## opens the .cshrc file in your root directory
cd ~/csci2270/hw2           ## move to this directory (if it exists)
```

The second is to specify directories relative to your current location in the directory tree. So if we are in the directory ~/csci2270/hw2, then we can locate the directory above ours by using '..':

```
ls ..                       ## lists everything in the csci2270 directory
emacs ../../.cshrc &        ## picks up the .cshrc file in our root directory
```

You can run some programs from any variety of directories because your .cshrc file defines where to find them as part of the path, which is why you can run g++ or emacs from almost anywhere. Other executable programs, like your csci2270 programs, will require you to be in your current directory. Running these programs requires you to either type './' before the program name, or else modify your .cshrc file to add the current directory ('.') to your path as we did in lab 4 (admittedly, this makes things a bit less secure).

REVIEW AND EXTENSION OF UNIX COMMANDS

You can learn more about how to get the most of these by asking for the manual page via **man** or **info** (you may have to try both):

```
man mkdir
info rm
```

To quit the man page and get back to the command prompt, type q.

pwd: print name of current (working) directory.
Just type

```
pwd
```

This is helpful if you get lost in your directory tree.

mkdir: create a new directory.
To create one relative to the root, say

```
mkdir ~/foo
```

To creat one relative to your current directory, leave off the ~/:

```
mkdir foo2
```

If you make a compound directory, like this:

```
mkdir ~/csci2270/lab7
```

you need to make sure that all the directories above it (like ~/csci2270) exist already, using the ls command.

rmdir: erase a directory
This follows the same rules as mkdir. If you delete a directory, it must be empty of files and subdirectories.

```
rmdir ~/foo
rmdir foo2
```

cd: change to a certain directory
To change to top directory:

```
cd ~ or just cd
```

To change to the directory just above the current one in the tree:

```
cd ..
```

To change to a directory specified relative to the root:

```
cd ~/csci2270
```

To change to a directory specified relative to the current directory:

```
cd lab7
```

ls: list files and subdirectories of a directory
To list a directory's contents relative to the root:

```
ls ~/csci2270
```

To list a subdirectory's contents relative to the current directory:

```
ls
```

list all files in a directory, in verbose format, including configuration files, which start with '.' (.emacs, .login, .cshrc):

```
ls –la
```

list the same files sorted by modification time:

```
ls -lta
```

rm: get rid of files
To remove most files (except those starting with '.'):

```
rm *
```

To remove a particular file:

```
rm csci2270/nonsense.cxx
```

To remove several files (headers and makefile), the wildcard '*' helps avoid extra typing:

```
rm *.h Make*
```

cp: copy a source filename to a destination directory (and possibly a different filename):
To copy a file in your root directory to a file with a different name:

```
cp ~/.cshrc ~/.cshrc_bak
```

To copy a file to another directory (with its original name):

```
cp ~/.cshrc ~/csci2270/lab7
```

To copy a file from a directory to your current directory ('.') under a new name:

```
cp ~/.cshrc .cshrc-gecko
```

If you're in your ~/csci2270/hw2 directory and you want to copy your hw1 Makefile to your ~/csci2270/lab7 directory (and both directories exist!)

```
cp ../hw1/Makefile .
```

mv: move (or rename) a file; note that the old file goes away!
To rename a file, move it from the old name to the new name:

```
mv .cshrc-gecko .cshrc-octopus
```

You can also move a file to a new directory:

```
mv poly1.cxx ~/csci2270/lab7/poly2.cxx
```

Most of the above commands can be made recursive—so they work in the directory you call them in plus its subdirectories and the subdirectories' subdirectories, etc. Depending on the command, you usually include an –R or –r. To recursively list all subdirectories from the root:

```
ls –R ~
```

And to recursively delete all files and subdirectories from a directory called ~/whatever, you can say:

```
rm –r ~/whatever (but be careful! this is a bit drastic)
```

MORE ON EMACS

Handy reference card (you'll need to zoom in) is at

http://www.cs.colorado.edu/~main/lab/refcard-emacs.pdf

C- here is the control key

M- here is generally (but not always) the Alt key

Full manual at

http://www.cs.utah.edu/dept/old/texinfo/emacs19/emacs_toc.html

A few of the basic commands:

Control-x Control-c exits emacs (if you haven't saved, you'll get a prompt)
Control-x Control-s saves your file (you may need to ok this or specify where)
Control-k deletes one line of text
Control-y pastes in test you cut or highlighted with the mouse
Control-g cancels a command (very useful when you get into trouble)
Control-s lets you search the current file for text
Control-_(control, shift, and the – key!) undoes the last thing you did

Many nice conventions are set in Professor Main's .emacs configuration file; for example, the following lines set up your C++ formatting:

```
;;; c++-mode
(add-hook 'c++-mode-hook
'(lambda()
  (local-set-key [13] 'c-return)          ;;; RET with automatic indent
  (local-set-key "\ep" 'indent-all)       ;;; Esc-p pretty-prints file
  (c-set-style "k&r")                     ;;; Kernihan & Richie's style
  (setq c-basic-offset 4)                 ;;; 4 spaces for indentations
  (c-set-offset 'substatement-open 0)     ;;; No indent for open bracket
  (c-set-offset 'statement-cont 0)        ;;; No indent for new stmts
```

```
                                        ;;; Needed in emacs 19
                                        ;;; because it thinks
                                        ;;; function definitions are
                                        ;;; new statements!
    )
)
;;;
```