**CSCI 4727 Homework 1**

This assignment is intended to give you some C/C++ and Linux practice and will be used as the basis for later assignments. Use the g++ compiler and a makefile on einstein.etsu.edu. ***This is an individual assignment.***

See the example code and makefile on einstein in my public directory

~harrisonms1/csci4727/examples

and the handout on Unix functions and make. If you need more details on Linux commands or system calls, read the manual page (for system calls, use the command man 3 mkdir, for example; the “3” says to look in section 3, which is where the functions are documented – without it, you’ll get the documentation on the Linux command line version of mkdir). You may need other Linux system calls, depending on how you solve the problems.

The program will accept the following pairs of command line parameters; the pairs are allowed in any order, but the pair itself must be in order (e.g., the –c pair can come before the –s pair, but –c must be paired with the command file name).

-s setupfilename \*Required\*: read set-up information from the file setupfilename

-c cmdfilename \*Optional\*: read commands from the file cmdfilename instead of

standard input; if missing, read commands from the keyboard.

-l logfilename \*Optional\*: name of the log file; if missing, use “log.txt”

See the example programs for how to process command line arguments.

After processing the command line arguments, create and open the log file in the current directory. Log records will have the format <timestamp> <data>. To get the timestamp, use time( ) and ctime( ); the latter returns a char\* string with the date, time, and a newline. Start each session a “Begin“ record. Open the required set-up file for reading; read this file, one line at a time, and write those same lines to the log file (again, with time stamps). For this assignment, don’t process the set-up commands – just copy them to the log file, then close the set-up file. Next, if the command file option “-c” was present, open that file for reading, read the commands, one line at a time, and write those same lines to the log file. Again, for this assignment, don’t process the commands – just write log records in the log file, then close the command file. If the command file option was not present, read commands, one line at a time, from the keyboard and copy them to the log file. In either case, the last command will be ‘Q’ by itself on a line. At that point, write a last log record, the timestamp plus “End” and close the log file.

Example: suppose that the file “setup” contains the lines

R 1

X 5

Y 5

and the file “command” contains the lines

M 1 N

M 1 E

Q

and you run this:

hw1 –s setup –c command

then the file “log.txt” will contain 8 records (16 lines, because the timestamps end with a newline):

<timestamp>

Begin

<timestamp>

R 1

<timestamp>

X 5

<timestamp>

Y 5

<timestamp>

M 1 N

<timestamp>

M 1 E

<timestamp>

Q

<timestamp>

End

where the <timestamp> will be the value returned from ctime(). If you run the program again with exactly the same input, the log file will have 16 records (32 lines), because you should \*append\* the new records.

The log file functionality will be based on the API contained in log.h; copy this file from ~harrisonms1/csci4727/examples and create log.cc (or .cpp) based on that. If the log file cannot be opened, display an error message and exit. If the set-up file is not present, write “Set up file “ setupfilename “ does not exist” and an End record to the log file (plus the required time stamps), and display an error message, and exit. Then if the –c flag is present but the command files does not exist, write “Command file “ cmdfilename “ does not exist” and an End record to the log file, display an error message, and exit.

Use the following wacky C++ trick to help with command file input – you’re supposed to read commands from a text file \*IF\* the –c flag is present, but read commands from the command line if it isn’t. The code below chooses between an input file infile and cin (standard keyboard input) so that all later input uses the same variable, input.

// Assume that, if the –c flag is present, you’ve already successfully

// done this: ifstream infile.open(cmdfilename)

istream& input = ( strlen(commandfile) > 0 ) ? infile : cin;

// Then later:

getline( input, str) ;

**Deliverables:** Your code ***\*must\**** execute correctly on einstein! Submit the source code (***no .o files or executables!***) and Makefile of your program to the Homework 1 dropbox in a zipped folder. Make sure that your Makefile names your executable “hw1” with the –o hw1 flag. Your zipped folder should be named <lastname><firstname>HW1.zip, e.g., HarrisonMatthewHW1.zip. I will slap you with a wet squirrel if I get a .rar archive (the squirrel smells pretty bad.) No hard copy is required.