These exercises are designed to familiarize you with C and the linux command line environment. You will work with gcc and an editor such as vim, emacs, or nano. Be sure to read/review Appendix F: Laboratory Tutorial of the textbook before you begin.

Please submit ALL PROBLEMS as attachments to ONE EMAIL to karavan@pdx.edu with subject: CS532 HW1 by 6pm on Tuesday January 14.

Problem 0: Vocabulary

Write short definitions for each of the following:

a) Operating System

b) manycore [include in your definition examples of 2 manycore processors other than GPUs]

c) Multiprogramming

d) Timesharing

e) POSIX

f) temporal locality

g) spatial locality

Problem 1: Using the Linux Shell

Use the `script` utility to record steps (1) - (11) below. Each step should be done with just one command line. Submit the file created by script.

[Look over the man page for the SCRIPT utility. You will use SCRIPT to record your work in this lab.]

1. Make a directory named Lab1Files
2. Change directories so that you are in Lab1Files
3. List the contents of Lab1Files
4. Copy all files from /u/karavan/public/LAB_01-1/ to your Lab1Files directory
5. List all the files in Lab1Files and show the access permissions
6. Change the access permissions on all of the files in Lab1Files so that the user has rwx access permissions, but group and other have no access permissions
7. List all the files in Lab1Files and show the access permissions
8. Display the contents of Lab1File1.in
9. Display the contents of Lab1File2.in
10. Execute this command line:

```
./changeIt Lab1File1.in Lab1File2.in
```

`changeIt` is a shell script that takes 2 command line arguments. Note the file permissions for `changeIt` must include permission to execute.

11. List the contents of Lab1Files and show the access permissions. **SUBMIT**: your script file.

Problem 2: Hello World (hello.c)

Compile the program hello.c using gcc. You will see an error message. Fix the error and compile again.

Use a compiler option to name the executable program "hello". [hint try man gcc to see the common options]

Copy hello.c to a new file, alien.c Edit alien.c to print out "hello, world 2" instead of "hello, world". [hint: try man printf to see the format string information you need for this change] **SUBMIT**: hello.c and alien.c

Problem 3: Math (tan.c)

File tan.c contains a not-quite-right version of a program to print out the tangent of 90. First, try to compile it with gcc. Then, fix the errors so that it compiles and runs correctly. [hint: try man tan]

SUBMIT: your corrected tan.c

Problem 4: Debugging basics

1. Copy the contents of the lab directory into your own home directory. (note: check the man page for cp and see the -r option).
2. Compile buggy.c with the command line option for gdb, into an executable named buggy.
3. Run buggy.
4. Oops! There is an error. Start gdb

   chinstrap:~/public/cs201_lab2> gdb buggy
   This GDB was configured as "x86_64-linux-gnu".
   Reading symbols from /u/karavan/public/cs201_lab2/buggy...done.

5. Now run buggy under the control of the debugger (gdb) run
6. The run stops at the error. Skip to the error line in the source file (if you don't know how to do this quickly in your editor, ask the tutor for help).
7. Examine the runtime error of p by entering at the gdb prompt: print p
8. Fix the error in buggy.c
9. Re-compile buggy, still using the gdb option
10. Re-run buggy to verify that the error is fixed.
11. Restart gdb with buggy
12. Run gdb
13. Use print to look at the value of x. Does it work?
14. Set a breakpoint at the start of main. (use gdb commands "list" and "break") This will stop the

    running program when it is about to execute the instruction that matches the source code line

    number.

15. Now run. The program will stop at the breakpoint. Now print out the value of x.

SUBMIT: There is nothing to submit for Problem 4.

Problem 5: printf()

1. Write a C program that prints out (using printf()) :

    - A string
    - An integer
    - A floating point number
    - A char

2. Write a makefile to build and run your program
3. Experiment with redirecting the output to (a) a new file (b) an already existing file

SUBMIT: your C program and your makefile