**CSE-443/543: High Performance Computing**

**Lab #01**

Max Points: 32

**Objective**: The objective of this exercise is to:

* Build experience with modifying code, compiling it with g++ at the command line, and creating and modifying batch shell scripts in bash
* Build experience measuring the performance of code on a Linux system
* Gain familiarity with the following Linux commands: nohup, cat, grep, uniq and /usr/bin/time
* Gain familiarity with the Linux /proc filesystem

Fill in answers to all of the questions in this document and enter data into the corresponding Google spreadsheet. You may discuss the questions with your instructor. To get full credit, make sure you follow the instructions carefully, including submitting exactly what is requested!

## Background

In this course, we will be looking at the performance of code and the effect that different actions can have on that performance. To quantify these effects, you will need to know how to perform various kinds of benchmarks on the code. The most straightforward benchmark is the amount of time (elapsed and/or CPU) the code takes to run.

Linux has (at least) two different timing commands available, one built into bash and one that is a standalone program. Timing the execution of a program using the time command invokes the bash version. To get the standalone program you must specify the path to the executable, /usr/bin/time. In this course, we will use /usr/bin/time rather than time because of the additional information provided by /usr/bin/time. You may want to check the [man page for time](https://linux.die.net/man/1/time) to get more information about it.

For this lab, you will be timing a program that computes pseudorandom numbers and then modify that program to compute some trigonometric functions (sine and tangent) in addition. You will use this timing data to estimate how many clock cycles the processor takes to perform these computations. The processor can perform some operations in a single cycle, but random numbers and trigonometric functions are more complex and take significantly more. How many? That is for you to determine.

In order to compute the number of clock cycles you need information about the speed of the processor. The [/proc](https://linux.die.net/sag/proc-fs.html) filesystem contains information about processes, devices and other aspects of the system. For example:

* The /proc/cpuinfo file contains detailed information about all of the processors (cores) in the system, and the “model name” line in the output shows the specifics of the CPU including its clock frequency.
* The /proc/meminfo file contains detailed information about the system memory and the “MemTotal” line in the output shows the specifics of the CPU including its clock frequency.

The base file for the lab, Lab01.cpp, generates 1,000,000,000 random numbers using the default built-in pseudorandom number generator rand(). You should compile this code and then run it five times, determining the average execution time via the /usr/bin/time command. Try timing a program like ls to see what the output looks like. You will be interested in the user time. You may find it helpful to use commands like grep or awk to isolate the user time and convert it to seconds (without a ‘:’ in it). You can also specify an output format for /usr/bin/time that will display the time without a ‘:’. Use the [man page for time](https://linux.die.net/man/1/time) to find out how if you wish.

Using this data, and the clock frequency you determine from /proc/cpuinfo, you can estimate how many clock cycles are required to compute one random number.

For subsequent experiments, you will modify the Lab01.cpp program to have it compute the sine or tangent of those random numbers. You will then estimate the number of clock cycles required to compute a sine and a tangent by determining the average run time for the modified program and subtracting off the base CPU (user) time (from just computing the random numbers).

In case it is not obvious, you can estimate the number of clock cycles for an operation in this way:

You need to make sure that the units are correct to get a result in cycles/instruction. As a hint: I made your lives a lot easier by choosing to have the program iterate 1,000,000,000 times!

## Experiments

1. Download the file Lab01.cpp from Canvas to your workstation.
2. Write a shell script called Lab01.bash that does the following:
   1. Invokes /bin/bash as an interactive shell (i.e. not a login shell)
   2. Prints a single line of “model name” information from /proc/cpuinfo
   3. Prints a single line of “MemTotal” information from /proc/meminfo
   4. Checks the Lab01.cpp code for errors using cpplint.py
   5. Compiles the Lab01.cpp file using our standard set of options and which names the executable Lab01.
   6. Runs the resulting executable five times, using the /usr/bin/time command to determine the CPU (user) time.
3. Run your shell script and use the output to fill in the “Apparatus” information below
4. Enter your observations in the “Observations – rand\_r()” table in your Google spreadsheet.
5. Modify the Lab01.cpp program to compute the sine of the result variable by replacing the line result += rand\_r(&seed); with the line result += sin(rand\_r(&seed));
6. Re-run your shell script and use the output to fill in the “Observations - rand\_r() and sin()” area of the Google spreadsheet.
7. Modify the Lab01.cpp program to compute the tangent of the result variable by replacing sin(rand\_r(&seed)) with tan(rand\_r(&seed)) in the appropriate place in the code.
8. Re-run your shell script and use the output to fill in the “Observations - rand\_r() and tan()” area of the Google spreadsheet.
9. Use the timing data and clock cycle information to complete the Analysis section of the Google spreadsheet.

# Apparatus (experimental platform)

## The experiments documented in this report were conducted on the following platform (fill in the Details column using information determined in your shell script):

|  |  |
| --- | --- |
| Component | Details |
| CPU Model | Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz |
| Main Memory (RAM) size | 8056076 kB |

# Submit to Canvas

When you complete the lab, download this document from Google Drive as a Microsoft Word (.docx) file with the naming convention Lab01 - MUid.docx (example: Lab01 - ferrenam.docx). You should save the corresponding Google spreadsheet file as a Microsoft Excel (.xlsx) file with the naming convention Lab01 Results - MUid.xlsx

Submit the following files to Canvas:

1. The Microsoft Word file you downloaded from Google Drive.
2. The Microsoft Excel file you downloaded from Google Drive.
3. The Lab01.bash shell script you created for this lab.
4. The last version of the Lab01.cpp program (i.e. the one where you computed the tangent).