CIS 3110: Operating Systems

Assignment 1: Threads and Processes

Due Wednesday, February 5, 2020 @ 11:59pm

1. Objective

The objective of this assignment is to familiarize you with the concepts of process and thread. For this first assignment, you won't be writing code; instead, you will be reading and executing it. In this assignment you are provided several programs. Your tasks are to read, compile and execute them, and observe and analyze the output. Write up brief answers to the questions posed in the lab. Place the write-up in a file called **answers-a1.txt**. Also, you are required to create a makefile with the rules for building the executables of these programs provided.

2. Experiments

Write up brief answers to the questions below. Place the write-up in a file called **answers-a1.txt**.

Experiment 0. Create a makefile (1 point)

Create a makefile with the rules for building the executables for the code provided. The name of the resulting executable is the name of code provided. For example, if you compile the code **thr_create.c**, the name of the resulting executable is thr_create.

Note that you assume all program source code is in the same folder as your makefile when you write it.

Also, please note that to compile and run a threaded C program (e.g., thr_create.c):

```
gcc -o thr_create thr_create.c -lpthread
```

Experiment 1. Thread and Process Creation (1 point)

Study the programs **thr_create.c** and **fork.c**. These programs measure the average time to create a thread using thr_create() and to create a process using fork(). Compile and execute the programs.

Q1) What do you conclude from this experiment? Briefly describe the reasons for the difference in timings.

Experiment 2. Process Management (1 point)

Study the program **proc_manage.c**. Compile and execute the program.

Q2) How many times "hello" is printed? Include the initial parent process, how many processes created by the program?

Note: you can redirect the output of the program to files, and then get count information on these output files using the wc command.

Experiment 3. Processes vs Threads (2 points)

Study the two programs **thr_shared.c** and **proc_shared.c**. These programs are identical except that one uses threads and the other uses processes. Compile and execute the programs.

Q3) What do you conclude from this third experiment? Explain the reason for the difference in behavior.

Experiment 4. The Counting Problem (1 point)

Study the program **ibadcnt.c** which creates two new threads, both of which increment a global variable called cnt exactly NITER, for example, by default NITER = 1,000,000. The variable cnt is initialized to zero. Therefore, the final value of the variable cnt is expected to 2*NITER, which is 2,000,000. Compile and execute it, and observe the output.

Q4) Is the program behaving "correctly"? What do you conclude from this experiment?

Please note that it is very important to run the programs multiple times in order to obtain the true behaviors of these programs.

Please note that all these programs are available in a compressed tar file a1_code.tar.gz which can be uncompressed using the following: tar zxvf a1_code.tar.gz. Also, an answer template **answers-a1.txt** is included in the compressed tar file.

3. Submission

You will submit the following files using the "DropBox" on the CourseLink.

1. A Makefile that compiles the programs

- make all: creates all the executables
- make clean: deletes the executable and intermediate files
- make zip: creates a zip archive with all your deliverables

2. answers-a1.txt that includes the following,

- Your name and student ID.
- Your write-up answers to the questions in the lab.

Presentation matters: your answer must be presented in a clean and concise manner that is easy-to-understand for others.

To hand in your A1, create a zip archive with all your deliverables and submit it on CourseLink. The filename must be cis3110_a1_XXX.zip, where XXX is your University of Guelph's email ID (Central Login ID). This naming convention facilitates the tasks of marking for the instructor and course TAs. It also helps you in organizing your course work. Failure to follow the requirements will result in mark reduction. Do not include any binary files in your submission.

You can submit your A1 multiple times and only the latest version will be kept and marked, so it is a good practice to submit your first version well before the deadline and then submit a newer version to overwrite when you make some more progress.

Note: to zip and unzip files in Unix:

```
$zip -r filename.zip files
$unzip filename.zip
```

Acknowledgements

This lab has incorporated the materials developed by Mirela Damian at Villanova University and Prof. Jim X. Chen at George Mason University. The copyright of these materials belongs to them.

References:
[1] Using make and writing Makefiles
https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_makefiles.html
[2] Compiling Programs with Make
https://web.stanford.edu/class/archive/cs/cs107/cs107.1202/resources/make
[3] POSIX Threads Programming
https://computing.llnl.gov/tutorials/pthreads/