CMPSC 400
Operating Systems
Spring 2020

Lab 3 Assignment:
Programming Processes and Associated Threads
Submit deliverables through your assignment GitHib repository bearing your name. Place source code in `src/` and written work in `writing/`.

Objectives

We will be writing code that creates multiple *parent* and *child* process-pairs. Each process is to launch two of their own threads. This is an open-lab, meaning that there is not just one correct way demonstrate how to combine processes and threads. Note, your program must run to completion and all threads are to be created, run and end.

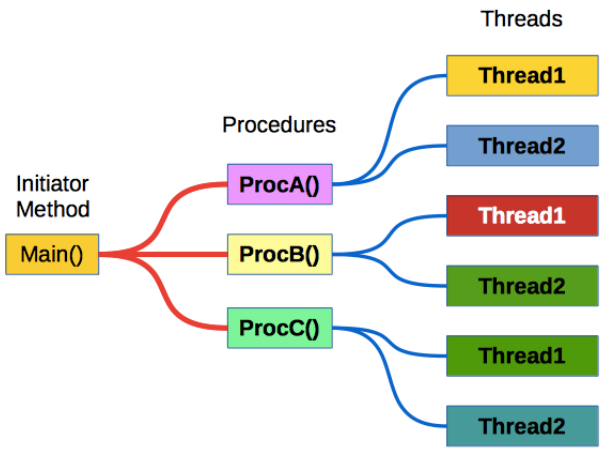

Figure 1: The flow for creation of threads from processes.

GitHub Starter Link

`https://classroom.github.com/a/ZzDT6q4L`

To use this link, please follow the steps below.

- Click on the link and accept the assignment.
- Once the importing task has completed, click on the created assignment link which will take you to your newly created GitHub repository for this lab.
- Clone this repository (bearing your name) and work on the lab locally.

Handed out: 4^{th} February 2020

- As you are working on your lab, you are to commit and push regularly. You can use the following commands to add a single file, you must be in the directory where the file is located (or add the path to the file in the command):
  - `git commit <nameOfFile> -m ‘‘Your notes about commit here’’`
  - `git push`

  Alternatively, you can use the following commands to add multiple files from your repository:
  - `git add -A`
  - `git commit -m ‘‘Your notes about commit here’’`
  - `git push`

Introduction

During our course this week, we have been working with *parent* and *child* processes. We have been working with writing C code to implement child processes from parent processes. We have also been writing code that creates and runs threads from a process. In this lab, we will put processes and threads together into one piece of code to demonstrate how to combine parent and child processes that (each) creates two threads, as shown in Figure 1. Although this code only has print statements, this code could be used to create many different (and diverse) processes which use threads to complete tasks.

By actively following the code and purposely breaking parts of code during development or testing, or by adding `printf()` statements in the code to show what is happening at each point, the reader will gain an understanding how the program works. Following the general structure of the code will provide the reader with a good idea of the C programming syntax of the code and its order of operations for creating threads from processes.

Extra Functionality From Libraries

Following your samples of thread-building code from class, you should have all the concepts in place to generate sophisticated code for producing processes and threads. Where possible, the reader is encouraged to consult the resources of the *inter-web* learn more about the following keywords which may help in your solution. Some of the following functions may be helpful. Note: In addition to these functions, there are other thread functions available in C that you could use (see `http://man7.org/linux/man-pages/man7/pthreads.7.html` for even more). Choose two or three to try in your code. In your questions, you will be asked which of these functions you chose to implement in your code and how they were used in your code.

- `fork()`
- `pthread_create()`
- `pthread_join()`

- `pthread_equal()`
- `pthread_exit()`
- `pthread_self()`
- `pthread_cancel()`
- `pthread_setcancelstate()`
- `pthread_setcanceltype()`
- `pthread_rwlock_rdlock()`
- `pthread_rwlock_timedrdlock()`
- `pthread_rwlock_timedwrlock()`
- `pthread_rwlock_wrlock()`

Project Description

In this lab, you are to write a new C program (called `src/procsAndThreads.c`) containing three methods (`ProcA()` ,`ProcB()`, and `ProcC()`) which are called by the `main()` method. Each of these three methods initiates a process which requires two threads to complete. In other words, on execution, each processes (`ProcA` ,`ProcB`, and `ProcC`) creates two of its own threads which must run and then complete, as noted in Figure 1. Each of these processes must create and initiate two of its own threads which print out details about what thread number they are, as well as which processes it was that created them. All this activity must be initiated by the `main()` method, also as shown in Figure 1.

There is one main requirement for your code; all processes and all threads must run and contain printf statements to show that they have run. As your code and output must show, your processes and threads must all terminate correctly.

0.1 Sample Output

After compiling and running your *procsAndThreads.c* C program which employs some of thread methods mentioned above, your output should resemble the following.

```
Program initialization
     Running: ProcA:
         +thread 1 from ProcA
         +thread 2 from ProcA
     Running: ProcB:
         +thread 1 from ProcB
         +thread 2 from ProcB
     Running: ProcC:
```

```
        +thread 1 from ProcC
        +thread 2 from ProcC
Program termination
```

Questions in Blue

1. How did you compile the code?
2. Generally speaking, how does the code work?
3. Which three library functions did you choose and how did you integrate them?

Required Deliverables

This assignment invites you to submit an electronic version of the following deliverables.

1. File: `src/procsAndThreads.c`: The working code for your program as detailed above and diagrammed in Figure 1.
2. File: `writing/report.md`: A report document to show the output of the program and to describe how it works to combine parent and child processes with thread generation. In your discussion, please be clear in describing how the general functions (listed above) may serve in your C code. *Please also mention in this report how you compiled and ran your source code.*

1 Honor Code

In adherence to the honor code, students should complete this assignment on an individual basis. While it is appropriate for students in this class to have high-level conversations about the assignment, it is necessary to distinguish carefully between the student who discusses the principles underlying a problem with others and the student who produces assignments that are identical to, or merely variations on, someone elses work. As such, deliverables that are nearly identical to the work of others will be taken as evidence of violating the Honor Code.