Assignment 4-1: Working with Pthreads in C++

Assignment Instructions:

- You must use the virtual environment that you set up in Exercise 1-1-3 for this assignment.
- Be sure to compile and test each program to be certain it works as expected. If you aren't sure how to compile and run a C++ program, refer to the Build and Execute Program section of the setup instructions document.

Important notes:

- At the top of your .cpp file, please include a comment with your **full name**. If your section uses **Lightweight Teams**, add the names of the teammates whom you worked with to the same comment.
- Add your own **individual comment** for each function / major portion of code that you add, briefly explaining what that part does.
- If you are asked to submit screenshots and your submitted screenshots do not match with your program's actual behavior, we will consider that to be a **violation of academic integrity** and pursue it accordingly.
- Make sure to **organize and format** your code in a consistent way.
- If you refer to any online resource to understand a concept, see examples of the use of a particular syntax, etc., add a comment **citing** that resource (i.e., specify website name and link).
- You must only submit **.cpp** files. If you have multiple .cpp files, upload them individually and **not** as a zip / compressed file.
- No screenshot(s) will mean no grade for this assignment.

Assignment Objectives:

- Practice the use of **POSIX threads** or **Pthreads** in a C++ program
- Practice the basics of Pthread creation in C++
- Practice the use of the `pthread_create` function in a program

Assignment Tasks:

The goal of the assignment is for you to practice and use functions related to **POSIX threads or Pthreads creation & handling** in Unix based OSs. An online version of the Linux manual can be found here:**http://linux.die.net/man/**.

For this activity, you will need to refer to the **pthreads section** of the Linux manual, available here:https://linux.die.net/man/7/pthreads

Another useful resource is the POSIX Threads Programming page at Lawrence Livermore National Laboratory by Blaise Barney [URL: https://computing.llnl.gov/tutorials/pthreads/]

If you need help with **navigating** the file system through a command line terminal, refer to this: http://linuxcommand.org/lc3_lts0020.php

Assignment Setup (0 points)

Note:

- *You need to use the terminal to compile and run the program. Do not use your IDE's GUI.*

1. You will need to download, compile, and execute a small program using your virtual environment.
2. Type the following command ***into the terminal window*** to pull the project repository from GitLab:
(You can also copy the command from the assignment instructions)

```
git clone https://cci-git.charlotte.edu/jbahamon/ITSC_3146_A_4_2.git
```

3. Change directory into the newly created directory (folder) named ITSC_3146_A_4_2.
4. Issue the following command to compile one of the programs:

```
g++ pthread_test_3.cpp -o pthread_test_3 -lpthread
```

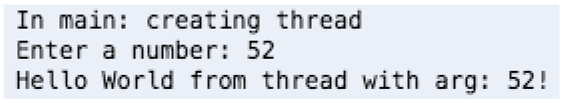
5. Issue the following command to execute the program:

```
./pthread_test_3
```

Part 1: Creating One Thread (2 points)

1. Take the **pthread_test_3.cpp** program and modify the **main** function, instead of pre-specifying/hard-coding the integer to be passed to the pthread function **PrintHello**, ask the user to enter an integer and pass that integer to the pthread function.

2. Build and run your program and make sure that it works correctly. Your output should be similar to the following:

   ```
   In main: creating thread
   Enter a number: 52
   Hello World from thread with arg: 52!
   ```

   Take a screenshot of your output and upload the picture as part of your assignment submission.

Part 2: Using Multiple Threads with The Same Data (6 points)

Your goal is to write a program that uses multiple threads to perform a series of calculations on the same data in the **pthreads_p2.cpp** file.

1. Take the **pthreads_p2.cpp** program and modify the ***main*** function to implement a loop that reads 10 integers from the console (user input) and stores these numbers in the **global** array named **arr** that has already been declared in the program (*this code will go right after the comment that says "Add code to perform any needed initialization or to process user input"*).
2. Modify the ***main*** function to create one ***pthread*** for **each** one of the functions that have been provided: countNegatives, average, and reverse (*this code will go between the comment that says "TODO: Modify according to assignment requirements"*).
3. Compile your program and run it several times. If the output of your program is garbled, you may need to add a small delay in between the multiple thread creation statements. One simple way to do this is to add a loop that performs a count, such as the one below:

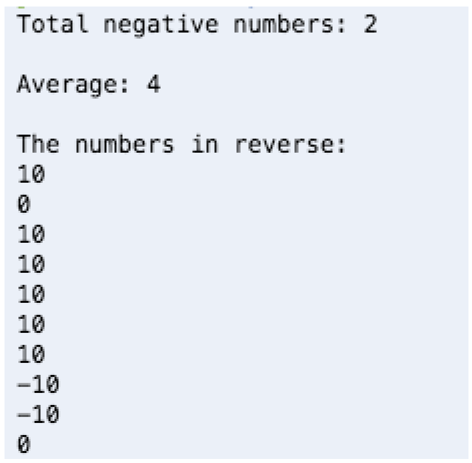
   ```
   for (int count = 0; count < 100000; count++);
   ```

   Note: You may encounter a warning that the for loop has an empty body, but in this case, you can disregard it.

Expected Output:

Given the following input: `0 -10 -10 10 10 10 10 10 0 10`

Your program should produce the following output:



```
Total negative numbers: 2

Average: 4

The numbers in reverse:
10
0
10
10
10
10
10
-10
-10
0
```

Take a screenshot of a sample output and upload the picture as part of your assignment submission.

Part 3: Creating Multiple Threads (10 points)

1. Make a copy of the program named **pthreads_skeleton.cpp**, which is also in the same directory, and name it **pthreads_p1.cpp**
2. Note the use of the **pthread_create** function in **main()**. Read the Linux manual page that describes the **pthread_create** function [URL: https://linux.die.net/man/3/pthread_create] and make sure that you understand the functionality that **pthread_create** provides and how to call (invoke) this function.
3. Modify the program to declare and initialize a **global array** named *my_messages* to the messages below:

   "Italian: Ciao!"
   "English: Hello!"
   "Hindi: Namaste!"
   "Spanish: Hola!"

   Use a `const char *` for this:

   ```
   const char* my_messages[4] = {"Italian: Ciao!",
        "English: Hello!", "Hindi: Namaste!",
        "Spanish: Hola!" };
   ```

   Note that you must use a character array just as shown here.

4. Modify the program to define a ***pthread function*** called ***printMessage*** that is sent a single integer parameter, ***index*** (wrapped in a void*). The function prints the message stored in the location ***index*** of the array ***my_messages***.

   You will need to use an ***explicit type cast*** *in the function to get the index value from the void* parameter.*

5. Modify the ***main*** function to:
   a. create as many ***threads*** as there are messages in the global array;
   b. instruct each thread to execute the function ***printMessage*** and pass each thread the ***index*** of a ***different*** location in the array as a parameter, using a **loop**;
   c. make sure to check for error conditions in thread creation.

   This code will go between the comment that says "TODO: Modify according to assignment requirements" and the "NOTE: Using exit here will immediately end execution of all threads" comment.

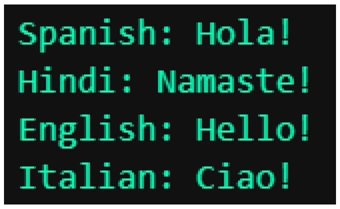
   Each thread will essentially execute the same function, but will print a different message based on the parameter it receives.

   Note: The order of execution of pthreads is not guaranteed, which means the thread that gets *created* first may or may not *execute* first. So, make sure the parameter you pass to a thread is guaranteed not to change during the execution of your loop.

6. Compile your program and run it several times. Note that the output of your program is likely to look garbled and may not always show the greetings in the same order. Run the program multiple times until you are confident that all of the threads are executing.

Expected Output:

Your program should produce the following output, but possibly with a different order of greetings:



```
Spanish: Hola!
Hindi: Namaste!
English: Hello!
Italian: Ciao!
```

Take a screenshot of your output and upload the picture as part of your assignment submission.

Submission

Caution: Before you submit, make sure that you have followed all the instructions under Assignment instructions and Important notes and that you have taken screenshots as indicated in the assignment.

Assignment Submission Items:

The files that need to be submitted for this assignment are the following

- pthread_test_3.cpp
- pthreads_p1.cpp
- pthreads_p2.cpp
- The necessary output screenshots for both cpp files.

Note: No screenshot(s) will mean no grade for this assignment.