CSE410: Operating System (Spring 2018)
Project #4: Dot Product using Single and Multiple processes/threads

Deadline

Thursday, April 5, 2018 at 11:59pm (Handin)

Assignment Goals

Get familiar with implementation of multiple processes/threads and their synchronizations.

Assignment Overview

In this project, you have to take care of producer-consumer problem when generating sequences of variables used in the later "dot product" operations. Also, you are required to design and implement a C/C++ program that does the "dot product" operations between two equal-length sequences of numbers by using single process/thread, multiple processes, and multiple threads methods.

Assignment Specifications

In this project you have to implement the following tasks (total 100 points).

1. **myDotProduct (5 points)**
   Your program will be invoked as follows.
   ```
   ./myDotProduct #ofValuesPerVector -option
   ```
   Let "myDotProduct" be the name of your executable. The program will generate and store two sequences of numbers and, then, do the "dot product" operation and print the result on the screen. Options (N, P, and T) are described as follows.
2. **–N (15 points)**
   Run the "dot product" operation in single process/thread mode.
3. **–P (20 points)**
   Run the "dot product" operation in multiple processes mode (2 processes). For this option, you are required to use the "mProduct" variable for storing the result in each process. Also, make sure that you do use the following concepts/function calls when dealing with this mode: fork(), shared memory, and semaphore (to avoid race condition occurs when saving the result to mProduct in each process). You could look at the skeleton code for more details.
4. **–T (20 points)**
   Run the "dot product" operation in multiple threads mode (2 threads). Again, you are required to use the "mProduct" variable for storing the result in each thread. Also, make sure that you do use the following concepts when dealing with this mode: pthread and mutex (to avoid race condition occurs when saving the result to mProduct in each thread). You could look at the skeleton code for more details.
5. **Generate 2 sequences of numbers (20 points)**
   The value specified in "#ofValuesPerVector" will be used for generating numbers for the 2 vectors. For example, if it is 6, your program should generate two sequences of numbers (value is within the range of 0 to 100) as follows.
   ```
   1 2 3 4 5 6   // vector 1 = [1, 2, 3, 4, 5, 6]
   9 8 7 6 5 4   // vector 2 = [9, 8, 7, 6, 5, 4]
   ```
   Moreover, to generate these values, you have to use the Producer/Consumer method (as shown in page 226). There should be one thread for producing random values (producer) between 0-100 and two threads (consumers) for consuming the values generated from the producer thread and storing it in each vector (each thread is responsible for one vector). The producer thread will only generate one value a time, and, after the value is consumed by either consumer thread, it will generate another value until the number of values generated equals to the total number of

values required (i.e. #ofValuesPerVector*2). For this part, you are required to use pthread_cond_t and pthread_mutex_t for synchronizing the behavior of the producer and consumer threads. You could look at the skeleton code for more details. Make sure that you do destroy those initialized semaphores and conditional variables after completing the task.

6. **Dot Product**

   Given two vectors, A and B, the dot product of these two vectors would be as follows.

   A = [a_1, a_2, a_3, a_4, a_5, a_6]

   B = [b_1, b_2, b_3, b_4, b_5, b_6]

   $A \cdot B = a_1*b_1 + a_2*b_2 + a_3*b_3 + a_4*b_4 + a_5*b_5 + a_6*b_6$

7. **Print on the screen (10 points)**

   After finishing the operation, your program should print the all the values in each vector together with the result on the screen as follows.

   ```
   Vector1 (V1) = [1, 2, 3, 4, 5, 6]
   Vector2 (V2) = [9, 8, 7, 6, 5, 4]
   V1 dot V2 = 119
   ```

8. **Clean up after yourself (10 points)**

   After finishing the execution of the program, please clean it up after yourself, which means you have to deallocate all the memory allocated by "new" (if any), detach shared memory created (if any), and destroy mutex/semaphore initialized if any.

Assignment Deliverables

A zip file named according to your NetID_project4 (e.g. john9999.zip where "john9999" is the NetID) containing the following files. ***Project4.cpp*** – the source code for the main program

***.cpp** – all the .cpp files we provided in "Project_Skeleton" folder together with those your created if any.

****.h*** – all the .h files we provided in "Project_Skeleton" folder together with those your created if any.

Makefile – make file for generating an executable "myDotProduct"

Readme – anything you want us to know

The code should work on "cse410.msu.edu". Please make sure your program is clear from any compilation and runtime errors before submission. Note that your project file should be submitted with specified file name via the "Handin" system.

Assignment Notes

1. You can find all the required source files by downloading the "Project4.zip" file. In this archive file, you can find the following items.
   - *.cpp
   - *.h

   Note: you need to use "-lpthread" when linking your program.
2. All the functions you have to implement for this project.
   - DotProduct::GenerateValues()
   - DotProduct::Producer()
   - DotProduct::Consumer()
   - DotProduct::~DotProduct()
   - DotProduct::NormalDot()
   - DotProduct::MultiProcessInitialize()
   - DotProduct::MultiProcessDot()

- DotProduct::ProcessDotOperation()
- DotProduct::MultiThreadDot()
- DotProduct::ThreadEntry()
- DotProduct::ThreadDotOperation()
- DotProduct::Print()

3. Some useful links
    - http://linux.die.net/man/3/pthread_cond_init
    - http://linux.die.net/man/2/shmget
    - http://linux.die.net/man/3/sem_init
    - http://linux.die.net/man/3/sem_wait
    - http://linux.die.net/man/3/sem_post
    - http://linux.die.net/man/3/sem_destroy
    - http://linux.die.net/man/2/shmdt
    - http://linux.die.net/man/2/shmctl
    - http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html
    - http://pubs.opengroup.org/onlinepubs/007904875/functions/pthread_mutex_init.html