CS 347M : Operating Systems Minor

Concurrent Programming : A Query Handling System
9 March 2016

1 Background

Pthreads is a threads facility in the POSIX standards for Unix and Unix-like operating systems. A very brief summary of some of the important features of this threads facility follows:

- *Pthreads:* These are threads in the POSIX standards. A pthread is created through a system call. System calls are available for a parent process to synchronize with child processes and query their states.
- *Mutexes:* A mutex is a semaphore meant to be used ONLY for mutual exclusion.
- *Condition variables:* A condition is basically an event. A condition variable is a variable declared to be a condition. It is a synchronization facility permitting synchronization between processes that await an event and that can cause an event. When a process causes an event associated with a condition variable, it performs a *signal* operation on the condition variable. A signal operation on a condition variable wakes one of the waiting processes, if any; however, unlike a semaphore, it does not have a value and a signal operation does not have any effect other than waking one of the processes currently waiting on the condition variable. A *wait* operation on a condition variable would block the process that performs the operation until the **next time** a signal operation is performed on it. (Signal operations performed on the condition variable in the past have absolutely no significance for any process that performs a wait operation.)

2 Reference Material

An excellent tutorial on the pthreads facility is available at the following URL: https://computing.llnl.gov/tutorials/pthreads.

3 Specification of the Assignment

You have to implement a query handling system.

3.1 Features of the Query Handling System

The query handling system consists of the following 3 components:

- A mother process that creates child processes called query_maker processes. The number of query_maker processes is controlled by a parameter *pnum*.
- A server process that is responsible for servicing queries as follows: It creates threads called query_handling threads for servicing queries. The maximum number of threads that should exist at any time is limited to the value of a parameter *tnum*.
- A shared memory area that is used to store queries made by query_maker processes until they are handled by query_handling threads. This area can hold *qnum* queries where *qnum* is a parameter.

Details of these three components are described in the following: (**Note:** In addition to the actions mentioned below, the entities will also have to perform additional actions for producing report(s), whose details are given later in this document.)

1. Each query is an arbitray but unique string of 10 charaters. (You may generate it in any manner you desire, or read it from a file.)
2. Each query_maker process has a loop. In each iteration of the loop
   (a) It makes a query and stores it in shared memory. If there is no space to store it in the shared memory, it blocks itself until there is sufficient shared memory available for storing its query and then stores the query in the shared memory.
   (b) Waits until the processing of its query has been completed.
   (c) Performs some arbitrary computation of your choice that consumes some CPU time
   (d) Makes a request to the kernel to wait for some duration of time of your choice.

3. The server process arranges handling of queries stored in the shared memory in **FCFS manner** as follows:

   (a) It creates a new query_handling thread and gives it the next query in shared memory, taking care that the number of threads that are actively handling queries never exceeds *tnum*.

   (b) A query_handling thread

      i. Prints a report containing its own id and the query given to it for processing.

      ii. Performs some arbitrary computation of your choice that consumes some CPU time

      iii. Makes a request to the kernel to wait for some duration of time of your choice.

      iv. Declares that it has completed servicing of the query given to it

      v. Terminates itself.

3.2 What You Should Implement

Write a C program that implements the Query Handling System using pthreads and values of the parameters *pnum*, *qnum* and *tnum*. Your program should

1. Ensure that the following three properties hold:

   (a) Correctness

   (b) Maximum concurrency

   (c) No busy waits.

2. Produce a report giving details of each query as follows:

   (a) Id of process that made the query.

   (b) Time when the query was entered in queue.

   (c) Time when the query's processing was completed.

Hint: Determine the synchronization requirements in the system by considering

- Interactions among query_maker processes, if any

- Interactions between query_maker processes and the server
- Interactions between the query_handler threads and the server.

You have to submit three sets of sample data

1. *Set 1:* Sample data that shows normal working of your program.
2. *Set 2:* Sample data that creates a situation in which the shared memory is full and a query_maker process has to be blocked.
3. *Set 3:* Sample data that creates a situation in which *tnum* threads are active, unprocessed queries exist in the shared memory and the server has to wait before creating a query_handling thread for handling the next query.

4 Work and Submission

- You are to work in groups of 2 persons.
- You must submit a single tar file. It should contain the following files
    1. A README file giving your names and roll nos, and a comment on the status of your submission.
    2. 3 files giving the sample data mentioned in this document.
    3. Your code.
- Submission date will be announced later.