Operating Systems Laboratory (CS39002)

Spring Semester 2020-2021

Assignment 5: Implementation of multiple producer-consumer system where producers create prioritized jobs

Assignment given on:	February 25, 2021
Assignment deadline:	March 11, 2021, 1:00 PM

Part 1: Implement a producer / consumer set of processes using shared memory, with the following specifications (25 marks)

a) Your program will first read the values of NP (number of producers) and NC (number of consumers), and create the required number of producer and consumer processes. It will also take the number of total jobs to run as an input.

b) Create a shared memory segment SHM, which is shared among all the producer and consumer processes spawned by your code. The shared memory segment will contain a priority queue of finite size (say, can hold 8 elements). It will also create a *job_created* counter (integer) and a *job_completed* counter (integer) in the shared memory.

c) Each producer process should generate a computing job, waits for a random interval of time between 0 and 3 seconds, and inserts the computing job in shared memory queue. After insertion, the producer will repeat the process. Each computing job is represented by a structure which contains the following elements at a minimum:

   i. process id for the producer

   ii. producer number

   iii. priority of the process

   iv. compute time

   v. job id.

   The priority of the process is a number between 1 and 10 and for each job, job id is a random integer between 1 and 100000, compute time is an integer between 1 and 4. The producer generates each of these three numbers randomly while creating a job.

d) While insertion, if the queue is full, the producer waits until space becomes available. It will also print the producer number, pid and details of the job generated (e.g. producer: 3, producer pid: 37, priority: ***, compute time: ***). Then the producer will increase the *job_created* counter by 1.

e) Each consumer process waits for a random interval of time between 0 and 3 seconds, retrieves the job with highest priority in the shared memory priority queue, removed the job and prints the job details on the screen mentioning the consumer number, consumer pid, producer number, producer pid, priority, and compute time (e.g. consumer: 2, consumer pid: 53, producer: 3, producer pid: 37, priority: ***, compute time: ***). Then the consumer will increase the job_completed counter and will sleep for "compute time" seconds. If the priority queue is empty the consumer process will wait till a job is inserted in the buffer.

f) The parent process (i.e., your code) will wait till both *job_created* counter and *job_completed* counter reaches a specified number of jobs (e.g., 10), output the time taken to run those specified number of jobs and then kill the process.

g) Use mutex/semaphores to prevent concurrent updation to shared variables leading to possible race conditions.

Part 2: Implement a producer / consumer set of threads (25 marks)

- Redo the assignment above with *threads* of the same process (using pthread library) instead of different processes. Note that, threads share address space, so you might need to use different mechanism for sharing and updating the queue across threads.
- You also need to submit a report and clearly explain what you did differently (algorithms, data structures, functions) for threads and processes and why.

Submission Guideline:

- You need to upload one zip file for the assignment named Ass5_<groupno>_<roll no. 1>_< roll no. 2>.zip. The zipfile should contain three files:
    - Ass5_<groupno>_<roll no. 1>_< roll no. 2>_process.c or .cpp (replace <groupno> and <roll no.> by your group number and roll numbers), and upload it as part of the zip.
    - Ass5_<groupno>_<roll no. 1>_< roll no. 2>_thread.c or .cpp (replace <groupno> and <roll no.> by your group number and roll numbers), and upload it as part of the zip.
    - Ass5_<groupno>_<roll no. 1>_< roll no. 2>_thread.document.txt (replace <groupno> and <roll no.> by your group number and roll numbers). The doc should describe what different techniques did you use between the process and thread-based implementation. and upload it as part of the zip.
- You must show the running version of the program(s) to your assigned TA during the lab hours.
- **[IMPORTANT]: Please make only one submission per group.**

Evaluation Guidelines:

Total marks for this assignment: 50.

We will check features of your code (including but not limited to) if correct number of producer/consumers are created, if the jobs are correctly created, and the consumers are executing those jobs correctly (i.e., sleeping for the right time).