

## Operating Systems Laboratory (CS39002)

Spring Semester 2019-2020

**Assignment 3:** Implementation of a multiple producer-consumer system using shared memory where producers create prioritized jobs

**Assignment given on:** January 27, 2020

**Assignment deadline:** February 3, 2020, 1:00 PM

Implement a producer / consumer system using shared memory, with the following specifications:

- 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) [BONUS]: use mutex locks to prevent concurrent updation to shared variables leading to possible race conditions.

### Submission Guideline:

- Create a single program for the assignment, and name it Ass3\_<groupno>\_<roll no. 1>\_< roll no. 2>.c or .cpp (replace <groupno> and <roll no.> by your group number and roll numbers), and upload it.
- 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)