BCSE – 3rd year – 1st Semester – 2022
Assignment – II
Operating Systems Laboratory

1. Design a CPU scheduler for jobs whose execution profiles will be in a file that is to be read and appropriate scheduling algorithm to be chosen by the scheduler.
   Format of the profile:
   <Job id> <priority> <arrival time> <CPU burst(1) I/O burst(1) CPU burst(2) ……. >-1
   (Each information is separated by blank space and each job profile ends with -1. Lesser priority number denotes higher priority process with priority number 1 being the process with highest priority.)
   Example: 2 3 0 100 2 200 3 25 -1 1 1 4 60 10 ….. -1 etc.
   Testing:
   a. Create job profiles for 20 jobs and use three different scheduling algorithms (FCFS, preemptive Priority and Round Robin (time slice:20)).
   b. Compare the average waiting time, turnaround time of each process for the different scheduling algorithms.

2. Create child processes: X and Y.
   a. Each child process performs 10 iterations. The child process displays its name/id and the current iteration number, and sleeps for some random amount of time. Adjust the sleeping duration of the processes to have different outputs (i.e. another interleaving of processes' traces).
   b. Modify the program so that X is not allowed to start iteration *i* before process Y has terminated its own iteration *i-1*. Use semaphore to implement this synchronization.
   c. Modify the program so that X and Y now perform in lockstep [both perform iteration I, then iteration i+1, and so on] with the condition mentioned in Q (2b) above.
   d. Add another child process Z.
   Perform the operations as mentioned in Q (2a) for all three children.
   Then perform the operations as mentioned in Q (2c) [that is, 3 children in lockstep].

3. Implement the following applications using different IPC mechanisms. Your choice is restricted to Pipe, FIFO:
   a. Broadcasting weather information (one broadcasting process and more than one listeners)
   b. Telephonic conversation (between a caller and a receiver)

4. Write a program for *p*-producer *c*-consumer problem, *p, c* >= 1. A shared circular buffer that can hold 25 items is to be used. Each producer process stores any numbers between 1 to 80 (along with the producer id) in the buffer one by one and then exits. Each consumer process reads the numbers from the buffer and adds them to a shared variable TOTAL (initialized to 0). Though any consumer process can read any of the numbers in the buffer, the only constraint being that every number written by some producer should be read exactly once by exactly one of the consumers.
(a) The program reads in the value of *p* and *c* from the user, and forks *p* producers and *c* consumers.
(b) After all the producers and consumers have finished (the consumers exit after all the data produced by all producers have been read), the parent process prints the value of TOTAL.
Test the program with different values of *p* and *c*.

5. Write a program for the Reader-Writer process for the following situations:
a) Multiple readers and one writer: writer gets to write whenever it is ready (reader/s wait)
b) Multiple readers and multiple writers: any writer gets to write whenever it is ready, provided no other writer is currently writing (reader/s wait)

Contd. From page 1:

BCSE – 3rd year – 1st Semester – 2022
Assignment – II
Operating Systems Laboratory

6. Implement Dining Philosophers' problem using Monitor. Test the program with (a) 5 philosophers and 5 chopsticks, (b) 6 philosophers and 6 chopsticks, and (c) 7 philosophers and 7 chopsticks

7. Write a program that will find out whether a system is in safe state or not with following specifications:

Command line input: name of a file - The file contains the initial state of the system as given below:
#no of resources 4 #no of instances of each resource 2 4 5 3
#no of processes 3 #no of instances of each resource that each process needs in its lifetime 1 1 1 1, 2 3 1 2, 2 2 1 3
The program waits to accept a resource allocation request to be supplied by the user or read from another file:
For example: 0 1 0 1 1 indicates that p0 has requested allocation of 1 instance of R0, R2 and R3 each.
Your program should declare the result:
(1) should this request be granted?
(2) if your answer is yes, print the safe sequence in which all remaining needs can be granted one by one and also grant the request. If the requesting process's need is NIL, the program internally releases all its resources. Go back to accept another request till all processes finish with all their needs.
Testing:

a. Generate possible request sequences of each process.
b. Each such sequence must satisfy the maximum requirements of the process.

Last Date for submission and viva: Respective groups: October 17th /18th , 2022