King Saud University Computer Science Department

CSC227: Operating Systems

Process Management

Instructions:

- You can work in a group of not more than 3 students
- Last day to submit the project is: to be announced later
- To get the marks you (group) must demonstrate the project and explains the implementation.

MULTIPROGRAMMING OS SIMULATION:

CPU SCHEDULING & MEMORY ALLOCATION

Introduction:

It is required in this project, to write a program in Java that simulates the behavior of the multiprogramming operating system and use java threads to implement job scheduler, CPU scheduler, and CPU Execution. At the end of the simulation, you are expected to output some statistics regarding the behavior of the system. In the following sections, we will introduce the hardware specification, the multiprogramming OS features and the jobs requirements.

Available computer system for the simulation has:

Hardware:

The computer hardware is assumed to have:

1. A RAM of size 1GB, where 320MB is used to store the OS.
2. A single core CPU that executes one instruction each unit of time.
3. An I/O device for input and output operations.
4. An internal clock that allows to measure time in milliseconds.

Operating System:

The operating system is the multiprogramming OS. We would be interested in only two features in this simulation: The Job and CPU scheduling.

1. Job Scheduling: The system implements multiprogramming batch processing.
2. A long term scheduler selects jobs in sequence and allocates them the needed memory until the 85% of the memory is full. A job is loaded only if there is enough memory to satisfy its first memory requirement.
3. Each required memory block should be contiguous but a process may have more than one block of memory in different locations of the memory
4. The short term scheduler will allocate processes to the CPU following **Shortest Remaining Time First (SJF)** sequence. Each process will remain in the CPU until the end of its CPU burst or a new shortest process enters the ready queue then it will perform an I/O burst for the requested period before becoming ready again.
5. A process will have several CPU-burst / IO burst sequences at described in the example below. In each CPU burst a process may require additional memory or decide to free part of its memory.

6. If a process requires additional memory and there is not enough memory to satisfy its request, it should be put in Waiting state until there is enough memory for it.
7. Any process that is put in WAITING state for I/O or memory allocation will be put at the end of the ready queue after the end of its waiting.
8. If all processes are in Waiting state, only if all waiting for memory allocation, this is a deadlock. The system should declare a deadlock and select the largest waiting process to be killed in order to get some free memory for the other processes.
9. At any moment, the processes will have one of the states, READY, WAITING, RUNNING, TERMINATED, KILLED.
10. When the job queue is empty and all processes are killed or terminated, the system should write a file containing statistics about all processes and their final status TERMINATED or KILLED.
11. Every 200 milliseconds, the long-term scheduler will wake-up, check the memory and load more jobs until the 85% of the memory is full.

Program specifications:
Each job in the jobs queue is defined as a sequence of several CPU-burst / IO burst as follows:

Job description	**Explanation**
Name	Process name
15 11	CPU-burst of 15ms – Memory required 11 MB
3	I/O burst of 3ms
5 4	CPU-burst of 5ms – Additional memory required 4 MB
6	I/O burst of 6ms
3 -6	CPU-burst of 3ms – Free 6 MB of the allocated memory
1	I/O burst of 1ms
3 0	CPU-burst of 3ms – No change in memory
-1	Job terminates after the last CPU-burst

You must generate these numbers randomly as follows:

1- **CPU-Burst range: 10-100**
2- **Memory size: 5-200**
3- **I/O Burst: 20-60**
4- **Arrival time: 1-80**

Initialization phase:
You should perform the following steps before running the simulation:
1. Load all jobs in the jobs file into a jobs queue in memory
2. Start the system clock (in milliseconds)
3. Start the long term scheduler that check the first job in the job queue, check if there is enough memory for it and then:
    a. create a process for that job,
    b. allocate its memory
    c. put it in the Ready queue,
    d. remove it from the jobs queue

4. Load the RAM with the maximum number of user programs, then go to sleep for 100ms.
5. Start the simulation run, which consists of a simulation of the Machine Execution Cycle. At each millisecond, the scheduler will check if a job CPU-burst has ended and if the I/O burst of a process has ended or a new process enters the ready Queue and its time is less than the remaining time of the currently running process (preemption). It should also check if any waiting process can be reactivated and put in the ready queue.

Output from the simulation:

A text file containing statistics about all processes and their final status TERMINATED or KILLED. Statistics about a process should contain:

a. Process ID
b. Program name
c. When it was loaded into the ready queue.
d. Number of times it was in the CPU.
e. Total time spent in the CPU
f. Number of times it performed an IO.
g. Total time spent in performing IO
h. Number of times it was waiting for memory.
i. Number of times its preempted (stopped execution because another process replaced it)
j. Time it terminated or was killed
k. Its final state: Killed or Terminated
l. CPU Utilization