Lab3
Scheduling

Objectives:

1. To get hands on experience with simulating few of the OS scheduling algorithms.
2. To better understand how scheduling algorithms are implemented.
3. Learn about how queues and files reading / writing are implemented in C.

Problem Statement:

You are required to write a program that simulates two operating system scheduling algorithms:
First Come First Served (FCFS) and Round Robin (RR). Your program reads an input file that lists the information of the processes that need to be scheduled to run on the processor. You need to simulate scheduling these processes using one of the two algorithms and output the state of each process at every cycle time. You will need to maintain statistics about the processes and output them after the simulation finishes.

After the program starts you will get 3 inputs in your program :

- 0 or 1 to select the scheduling algorithm to use in your simulation. 0 refers to using First Come First Served (FCFS). 1 refers to using Round Robin (RR).
- quantumTime: this is the quantum time to use in simulating the Round Robin algorithm. This must be a positive number that is greater than zero. Note that this number should only be present if the first input is entered as 1 (RR).
- inputFileName: the input file with the processes and their arrival and burst times.

Each line in the input file contains four numbers interpreted as follows :

- Process ID
- CPU time : this is the time the process needs to execute on the CPU to complete.It is represented as a number of simulation cycles.
- I/O time : his is the time the process will need to wait for I/O. Note that this is the number of simulation cycles a process will remain blocked, and you can safely assume that processes do not compete for the I/O devices.
- Arrival time:this is the time the process becomes available in the system.It is represented as simulation time cycle.

A process arriving at the system will run as follows. It will need to run for CPU time, block for I/O time, and then run for CPU time again. Note that all the numbers are integers. Few lines from an example input file are shown below:

0 1 2 0
1 1 1 5
2 1 1 3

In the above example, process 0 will run for one simulation cycle, then block for two simulation cycles, and finally run for one simulation cycle. The process might wait in the ready queue before being scheduled on the CPU to run. Note that a process added to the list of processes that are blocked for I/O can take any time in this list, and does not have to be

removed from it before other processes added to the same list after it. In other words, it is a bad choice to simulate the list of blocked processes as a FIFO queue.

Simulation Output :

You should output a text file.You should call it ouputFCFS.txt or outputRR.txt the output file will be formatted to output two parts .

The simulation of the scheduling starting from simulation time cycle 0. Every line shows the following:

- Simulation cycle time.
- The state of each process: Running, Ready, or Blocked. This should appear as follows: K: STATE to represent the state of process with ID K. Therefore, for a running process 1,you should present it as 1: Running.
- Note that at simulation listing, you should not type process that have terminated or that have not arrived yet.

Statistics collected about the simulation. You need to print the following:

- Finishing time. This is the last simulation time at which all the processes have terminated.
- CPU utilization. This can be calculated as the ratio between the number of the simulation
- cycles at which the CPU was busy divided by the total number of the simulation cycles.
- For each process, print the turnaround time. This is calculated as: (the cycle the process finished - the cycle the process arrived +1).

An example output file looks like the following given the input information in the previous section and after running FCFS algorithm:

0 0: running
1 0: blocked
2 0: blocked
3 0: running 2: ready
4 2: running
5 1: running 2: blocked
6 1: blocked 2: running
7 1: running

Finishing time: 7
CPU Utilization: 0.75
Turnaround time of Process 0: 4
Turnaround time of Process 1: 3
Turnaround time of Process 2: 4

Scheduling Algorithms

You will need to simulate the following two scheduling algorithms.

First Come First Serve (FCFS):

You will need to note the following:

1. You need to maintain a queue of ready processes.
2. When a new process arrive to the system (read from your input file), you need to add it at the end of the queue. You need to note that the arrival time of this process need to be equal to the simulation time to consider adding the corresponding process to the queue.
3. Processes that finish their I/O blocking time are added to the end of the queue.
4. If two processes are ready at the same time, the one with the lowest ID is chosen to run.
5. If two processes arrive at the same time, then the process with the lowest ID is added to the ready queue first.

Round Robin (RR):

You will need to note the following:

1. The scheduler selects a new process to run on the CPU in the following cases:
    a. The process currently running on the CPU finishes its CPU time and it will terminate.
    b. The process currently running on the CPU finishes its CPU time and it will block for I/O.
    c. The process currently running on the CPU has not finished its CPU time. However, it has spend number of cycles on the CPU equal to the quantum time.
2. You will need to maintain a queue of ready processes. When a process is preempted before it finishes its CPU time, it is added to the end of the queue.
3. If two processes are ready at the same time, the one with the lowest ID is chosen to run.
4. If two processes arrivat the same time, then the process with the lowest ID is added to the ready queue first.

Deliverables:

- Complete source code, commented thoroughly and clearly.
- A report that includes:
    - A description of the overall organization of your code and the major functions.
    - Sample runs and screenshots.

Notes:

- Languages used: C/C++.
- Operating System: Linux
- Students will work in groups of 4-5.
