



EE 441 HOMEWORK #1

Processor Simulation

Due: November 11, 2019, 23:59

For questions: ferhatg@metu.edu.tr

You are going to simulate the processes running on a processor. Each process will have the following features: ID, Priority, Arrivalttime, Runtime. Runtime is an integer which shows the number of cycles of the remaining execution time of the process.

The time unit in the simulator is cycles. Arrivalttime and Runtime are in the unit of cycles of execution. Priority is indicated by an integer which can be 1, 2 or 3. Smaller value indicates a higher priority.

The simulator keeps track of the object Running_Process of type Process which shows the current process in execution. If there are no processes in the system, the ID of the Running_Process is -1.

When a process starts execution in the processor, it is copied in the Running_Process and its Runtime is decremented in each cycle. When Runtime is 0, the process finishes the execution. The latency of each process is defined as follows: Finish time-Arrival time

If a new process arrives at an idle system (the processor is idle, process queue and process stack are empty) it starts execution immediately.

We need to store the arriving processes if the processor is already executing a process. There are 3 separate process queues for each priority that store the arriving processes that we call Q1, Q2 and Q3. There is a single process stack that stores the preempted processes.

When a new process with priority i (P_i) arrives, if there are any processes in the system (in the queues, on the stack or in execution) with a higher or equal priority, P_i is inserted in Q_i .

If during the execution of a process a higher priority process arrives, then the current process in execution is preempted and pushed on the process stack as explained with an example scenario below.

When the processor finishes the execution of a process, it should select the highest priority process in the system as the next process to execute. If there are two highest priority processes, it selects the older one.

Example: Assume the system is idle at $t=0$.

At $t=0$: Process P1 arrives with Priority=3 and Runtime=50. P1 Starts execution.

At $t=10$: Process P2 arrives with Priority=2 and Runtime=70. P1 is preempted and pushed onto the stack with Runtime=40. P2 starts execution.

At $t=20$: Process P3 arrives with Priority=3 and Runtime=30. Stored in Q3.

At $t=30$: Process P4 arrives with Priority=1 and Runtime=10. P2 is preempted and pushed onto the stack with Runtime=50. P3 starts execution. Stack has P1 and P2 (top).

At $t=40$: P4 completes. Processor has to select the next process to run among P3 in Q3 and P2 on stack. Selects P2 (higher priority). P2 resumes execution with Runtime=50.

At $t=50$: P5 arrives with priority 2 and Runtime=30. P2 is not preempted as it has the same priority and it is older. P5 is stored in Q2.

At $t=60$: P6 arrives with priority 1 and Runtime = 10. P2 is preempted with Runtime=30. P6 starts execution.

At $t=70$: P6 completes. Processor has to select the next process to run among P3 in Q3, P5 in Q2 and P2 on stack. Selects P2 (higher priority than P3, older than P5). P2 resumes execution with Runtime=30.

At $t=100$: P2 completes. Processor has to select the next process to run among P3 in Q3, P5 in Q2 and P2 and P1 on stack. Selects P5 (higher priority).

Input and output files for the example scenario are distributed with the homework. More examples for selecting the next process to run:

- 1) If Q1 has a process and top of the stack has priority 2, Q1 is dequeued and process starts execution.
- 2) If Q1 is empty and Q2 has a process and top of the stack has priority 1, stack is popped and the preempted process resumes.
- 3) If Q1 is empty and Q2 has a process and top of the stack has priority 3, Q2 is dequeued and process starts execution.
- 4) If Q1 has a process and top of the stack has priority 1, the stack is popped and the preempted process resumes.

Note that the latency includes the Runtime and all possible additional times the process spends in the queue and in the stack.

Before the simulator starts to run, it takes a text file (input.txt) as input which describes the process arrivals in the following format:

ID Priority Arrivaltime Runtime

where ID, Priority, Arrivaltime and Runtime are integers. Each line contains the information for one process. Simulation runs until the execution of all processes in the input file are completed.

You are going to provide a log file in the form of .txt with all arrival, finish, push, pop and preemption events in the simulation, clearly indicating the time the events occur. See the example file for the output format.

You will report the latency for each process ID on a separate .txt file. This file should be sorted in terms of processes' finish order.

You need to implement your own Stack and Queue classes. You can use the codes in the lecture notes. At any time of the simulation, there could be up to 1000 processes in each queue. That is, your queues should be able to store at least 1000 elements. (For the stack, maximum number of elements is 2.)

Hint: For file input/output operations, you might find `file_io.cpp`, which is distributed with the homework, helpful. Note that this code is not a complete solution to the homework. You will need to significantly change it according to your needs.

Regulations:

1. You should insert comments to your source code at appropriate places without including any unnecessary detail. Comments will be graded.
2. Use **Code::Blocks IDE** and choose GNU GCC Compiler while creating your project. Name your project as "e<student_ID>_HW1". Send the whole project folder compressed in a rar or zip file. You will not get full credit if you fail to submit your project folder as required.
3. Your C++ program should follow object oriented principles, including proper class and method usage and should be correctly structured including private and public components. Your work will be graded on its correctness, efficiency and clarity as a whole.
4. Late submissions are welcome, but penalized according to the following policy:
 - 1 day late submission: HW will be evaluated out of 70.
 - 2 days late submission: HW will be evaluated out of 50.
 - 3 days late submission: HW will be evaluated out of 30.
 - 4 or more days late submission: HW will not be evaluated.

Good Luck!