CS260 - Programming Assignment 4 - Process queue simulation

The purpose of this assignment is to give you some practice working with queues. And you'll get some experience creating a simulation. Super fun!

Overview

A multi-tasking operating system such as Unix or Windows allocates CPU time to each of the processes in memory using a short-term scheduler.

A short-term scheduler works basically as follows:

> Processes are added to the queue with a priority level. The operating system removes the highest priority process from the queue and executes it for a short amount of time (such as 1ms, or maybe as much as 100ms). This time period is referred to as a tick, time-slice, or time quantum. If the process has not completed executing but the end of the quantum, it's interrupted and re-inserted back into the queue. The next process is removed from the queue and executed. There is always one process called the system-idle or null process. This gives the computer something to do when there are no other processes to executed.

For this assignment you'll write a program that simulates a process scheduler; specifically a (simplified) multilevel feedback queue (Windows, Linux, and OS X have all made use of this).

Background on Schedulers and the multilevel feedback queue can be found here:

- http://en.wikipedia.org/wiki/Scheduling_(computing)
- http://en.wikipedia.org/wiki/Multilevel_feedback_queue

Our simulator will be simplified as follows:

- we'll only have 10 priority levels. (10 queues!)
- processes will never voluntarily relinquish control (normally, processed might do that when waiting for disk I/O or availability of some other shared or slow resource)

Operational Details

Information about processes to run will come from an input file (described below). For our simulation, we'll just assume that every process arrives 15ms after the previous one. Process IDs will be assigned in the order they arrive (First one is PID 1, Second is PID 2, etc..).

While there are processes not yet complete.

> If there is a new process to be run, insert it into the proper queue
> Remove the highest-priority available process from the queue and run it for one timeslice (10ms)
> If it's done, output its statistics

If it's not done, lower its priority by one and re-insert it into the proper queue.

Technical Details

- You must prompt the user for the filename (until they enter a valid one).
- You'll need a struct or class to hold information about each process
- You can use a queue from the stl
- Ideally, your program should be easily configurable. Things like the timeslice and highest priority level should be constants.

Input

The input file will consist of multiple processes, one per line. Each process consists of the the following three pieces of information

- initial priority (from 0 (high) to 9 (low))
- required time (integer for total time the process will need (in ms)

Example Data Files

5 50	5 15	0 80
7 100	5 10	0 100
1 100	5 100	0 34
	5 40	0 28
		0 90
		0 75
		0 20

Output

As each process completes, output its

PID, initial priority, and total time needed (data from input file)
its final priority
the time it was submitted (in ticks, with the beginning of the program being time 0)
the time it completed

Very Important Stuff

Program should be well written, function properly, and be both easy and efficient to use.

All programs should follow the class's Coding Conventions
Submit the following:

- A zip file containing all your .cpp and .h files and your executable
- The executable as a separate file (so the .exe gets uploaded twice)