COSC 3360/6310—Operating System Fundamentals
Assignment #1 for Summer 2017: Process Scheduling
Due on Tuesday, June 27 at 11:59:59 PM

Objective

This assignment will introduce you to process scheduling.

Specifications

You are to simulate the execution of a stream of interactive processes by a time-shared system with a very large memory, a ***multi-core*** processor and one disk drive.

For simulation purposes, we will assume that each process consists of a fixed number of process steps that are known a priori. Each process will be described by its start time and a sequence of resource requests.

Input Format: Your program should read its input from **`stdin`** (**`C++ cin`**) and use input redirection as in:

```
$ assignment1 < input1.txt
```

This input will look like:

```
NCORES 2     // system has two cores
NEW    0     // new process starts at t = 0 ms
CORE   200   // get 200 ms
DISK   0     // do blocking disk access
CORE   30    // request 30 ms of CPU time
DISPLAY 100  // write to display for 100 ms
CORE   10    // request 10 ms of CPU time
INPUT  900   // wait for user input for 900 ms
CORE   10    // request 10 ms of CPU time
DISK   1     //do non-blocking disk access
CORE   30    // request 30 ms of CPU time
NEW    100   // new process starts at t = 100 ms
CORE   40    // request 40 ms of CPU time
...
```

but without the comments. Each process will execute each of its computing steps one by one and in the specified order. In addition, the start times of all processes will always be ***monotonically increasing.***

Disk Accesses: Disk accesses will either be ***blocking*** (code **1**) or ***non-blocking*** (code **0**). A blocking disk access puts the requesting process in the **`BLOCKED`** state until the disk access completes. A non-blocking disk access immediately returns the requesting process to the end of the **`READY`** queue without waiting until the disk access completes. All disk accesses will take 10 ms.

Memory Allocation: We assume that memory is large enough to contain all processes.

CPU Allocation: Your program should maintain a single ready queue for all processes and manage it in strict first-come first-server order (FCFS).

Disk Allocation: Your program should maintain a single FCFS queue for all processes waiting for the disk.

Input and Display Access: We will assume that each process will run in its own window so there will never be any queuing delay.

Output Specifications: Each time a process terminates, your program should output a short report with:

1. The total simulated time elapsed;
2. The current number of busy cores
3. The sequence numbers of the processes that perform a `CORE` step, a `DISK` step, a `DISPLAY` step or an `INPUT` step;
4. The contents of the ready queue and the disk queue;
5. For each process in main memory and the process that has just terminated, one line with: the sequence number of the process, its start time, the number of disk I/Os it has performed (including the current one) and its current status (**`READY`**, **`RUNNING`**, **`BLOCKED`** or **`TERMINATED`**);

Error recovery: Your program can assume that its inputs will always be correct.

Implementation

Your program should start with a block of comments containing your name, the course number, and so on. It should contain functions and these functions should have arguments.

It should have a process table containing all processes that have been loaded into main memory and have not yet terminated. This table should be used to keep track of process statuses and should be distinct from the data structure(s) that you might use to store your input data.

All times should be simulated.

Since you are to focus on the scheduling actions taken by the system you are simulating, your program will only have to intervene whenever

1. A process is loaded into memory,
2. A process completes a computational step.

You should not worry about minor ambiguities that could result in slightly different correct answers. Your program will be tested on inputs that produce unambiguous results.

These specifications were updated on ***June 13, 2017.*** Check the course web site for corrections, precisions and updates.