Lab Assignment 3 (Sub:OS)

Assignment ID: LA3

Anshumaan Agrawal (IMT2014007)

September 11, 2016

Execution and Compilation

Created a C++ program called \simulator.cpp" which simulates short term CPU scheduling in an operating system. The following scheduling algorithms are used. There is also context switch time added. To compile, just run the make le : "make all". To execute, just run this on the command terminal : "./simulator ".

First Come, First Served (FCFS)

Shortest-Job-First (SJF), with no preemption

Priority with no preemption (Priority)

Highest Response Ratio Next(HRRN)

Round-Robin (RR), with configurable time slice time_q

Multi Level Queue(MLQ)- The number of queues is taken from the user via config file

Multi Level Feedback Queue- The number of queues is taken from the user via config file

Input Format

There are sample text with input format :
(Number of Processes=NOP)
(Number of I/O involved= NOIO)
(Minimum Priority) (Maximum Priority)
(pid) (arrival time) (priority) (Number of I/O interuption + 1 = N) (Type) (Time required) (Type) (Time required) .....N times
.
.
.
.
NOP times
(time quanta for RR)
(Number of queues for MLQ and MLFQ)
Here: Type is an interger, 0 stands for CPU and 1-NOIO stands for I/O 1-NOIO respectively.
For example 1 stands for I/O1 and 2 stands for I/O2. 0 stands for CPU.

The inputs that are given- "test.txt", "test2.txt" and "test3.txt" and the corresponding graph samples "graph.png", "graph2.png" and "graph3.png".

Output Format

The output is given in the format where every policy is applied on the input le and the necessary calculations are done and then the best policy for the input format is displayed.

Format :

(Process id) (Arrival time in the queue) (Type of the queue)
.
.
.
TAT Total : (Sum of all the TATs)

(The best policy for the given input is printed out on the screen)

Code Structure

A "process, Set, Stat" class has been made. For each class, public functions and private variables have been de ned.

For each OS policy, separate functions have been written.

General Algorithm for all policies:-

```
current_time =0;
while(Process_Completed!= Number of Process) {
        Update new_queue (pushing new process to ready and blocked queue of respective I/O);
        Update ready_queue (choosing process according to the policy and running it for 1 unit);
        Update Blocked_queue;
        current_time++;
}
```

A "main" function is written for taking inputs from the file to a “Y vector".

A "statistics" function is written to calculate the turnaround time, wait-ing time and others for each OS policy.

At the end of the code, the best OS policy is displayed for the input file.