CS232 Operating Systems
Assignment 02: Simulate a scheduler
Due : Oct 12, 2023 @ 11:59PM

CS Program
Habib University

Fall 2023

1 Objectives

1. Refresh your C programming.
2. Understand process scheduling.

2 Hackerrank Link

In this assignment you will write a simulator for a process scheduler using C language. A hackerrank test is created on this link:

https://www.hackerrank.com/test/e9aegt8lkf/0b0c93b974c90ca83363fe7bc7487777
The link contains detailed instructions which are also copied below.

3 Description

3.1 Introduction

In this assignment you will write a simulator for a process scheduler using C language. You need to create a queue data structure along with code for parsing and filling the ready queue. Then you need to implement the four scheduling algorithms (FIFO, SJF, STCF, RR) by providing the body of the following four functions.

```
void sched_FIFO(queue *const pqueue, int *p_time){}
void sched_SJF(queue *const pqueue, int *p_time){}
void sched_STCF(queue *const pqueue, int *p_time){}
void sched_RR(queue *const pqueue, int *p_time){}
```

In all of these functions, the first parameter `pqueue` is the queue filled with all processes' control blocks. The second parameter `p_time` is the system time that is initialized with 0 and will be incremented in each step to simulate running of a process from the queue in your scheduling function.

3.2 Input

The first line will be the total number (N) of processes (int) and the scheduling policy (string) which could be one of FIFO, SJF, STCF, RR. Then, N lines of inputs will be given which will contain the following data separated by colon (:). The process name (pname), the process ID (pid), process total runtime (duration) and process arrival time (arrivaltime). These will be given in one line separated by colon (:) like this pname:pid:duration:arrivaltime. Pname will be a string of up to 10 chars. All other fields will be integers.

Sample Input:

```
3
RR
P1:12:7:3
P2:15:3:5
P3:1:6:2
```

3.3 Output

When your program simulates the scheduler, at every step, it should output the state of the system in the following colon-separated format:

`time:running name:ready queue names comma separated:`

where,

- `time` represents the number of clock-ticks passed since the system started. Assume a clock-tick lasts 1 millisecond. The system starts at 0, and the clock-tick between 0 and 1 ms is represented as 1 in the output.
- `running name` represents the name of the process that was in the running state for this clock-tick. If in a given clock-tick no process is running, the output shows the string `idle` here.
- `ready queue names` is a comma-separated list of names of processes in ready state during this clock-tick, with their time-to-completion in parentheses. In case the queue contains no processes, the output should display the string `empty`.

Sample Output:

```
1:idle:empty:
2:idle:empty:
3:P3:empty:
4:P3:P1(7),:
5:P1:P3(4),:
6:P3:P1(6),P2(3),:
7:P1:P2(3),P3(3),:
8:P2:P3(3),P1(5),:
9:P3:P1(5),P2(2),:
10:P1:P2(2),P3(2),:
11:P2:P3(2),P1(4),:
12:P3:P1(4),P2(1),:
13:P1:P2(1),P3(1),:
14:P2:P3(1),P1(3),:
15:P3:P1(3),:
16:P1:empty:
17:P1:empty:
18:P1:empty:
```

3.4 Output Performance Metrics

Once all test cases are passed on hackerrank, modify your code to additionally generate in the output the following three performance metrics:

1. Average throughput of the scheduler

2. Average response time of the scheduler
3. Average turnaround time of the scheduler

3.5 Compile the Performance Metrics of Scheduling Algorithms

You are required to run each of the four scheduling algorithms (FIFO, SJF, STCF, RR) on all five sample test cases provided on hackerrank. In your PDF submission, report the above metrics for all scheduling algorithms on each of the five sample test cases provided.

3.6 Compare the Performance of Scheduling Algorithms

Calculate the overall average throughput, response time, and turnaround time of each scheduling algorithm over all 5 test cases. Report your findings. Also highlight: Which algorithm performed best overall in regards to throughput? Which algorithm performed best overall in regards to response time? And which algorithm performed best overall in regards to turnaround time?

3.7 Assumptions to Keep in Mind

- Context switch is instantaneous.
- Newly arrived processes are inserted at the end of ready queue.
- If two processes arrive at the same time, then they should be inserted in ready-queue in the order they are found in the input.
- if at the end of time slice P1 is getting de-scheduled and P8 and P9 arrive at the same moment, P1 is inserted first at the end of the ready queue and then P8 and P9 are inserted.
- SJF and STCF keep the ready queue sorted at all times (w.r.t time-to-completion).

4 Submission

You will be required to submit:

- all the code (.c) files
- a makefile containing rules for build, rebuild, run
- a PDF file cointaining your name, ID, and sections for each .c file and the makefile containing their contents. Code contained in PDF should be displayed formatted and not exceeding page boundaries.
- the PDF file should additionally contain reports on performance metrics as required by the preceding sections

Marks:

- Proper parsing of the input data: +5 marks
- Proper implementation of queue +15 marks
- FIFO + SJF working perfectly: +20 marks
- STCF + RR working perfectly: +30 marks
- General working: +10 marks
- Performance metrics: +10 marks
- Submission conforms to guidelines: +10 marks

Penalties:

- Code doesn't compile: -100 marks
- Code has warnings: -10 marks
- Program crashes: -30 marks
- Late submission: -20 marks + -10*num_days

Marks obtained = max (0, marks + penalties).