HW4: Programming Assignment v1 30Sept.2021.8:00PM

CPU SCHEDULING ALGORITHMS

In this assignment you will implement three scheduling algorithms to generating the Gantt charts and evaluate measures of effectiveness.

Due Date: Thursday, October 14, 2021, 11:00 pm

Extended Due Date with 20% penalty: Friday, Oct 15, 2021, 11:00 pm

1. Description of Task

In this assignment, you will be implementing the three CPU scheduling algorithms mentioned below. The first command line argument specifies the file name which contains the list of processes for your scheduling algorithms. The next command line argument specifies the time quantum (for Round Robin). Implement the following three CPU scheduling algorithms using either C or Python 3 programming language (you are free to use Pandas and NumPy libraries if needed). Assume that all jobs are CPU bound (i.e., they do not block for I/O), and the context-switching time is negligible. Also, assume that in Round Robin if a new process arrives the same instant when a process is switched out, the new process gets in the ready queue first.

- First Come First Serve (FCFS)
- Priority Scheduling (PS) without pre-emption
- Round Robin (RR) with the specified quantum.

You must produce a Gantt chart for each scheduling algorithm to help visualize the sequence of execution for each process (**See example output below**). You will analyse the performance of these scheduling algorithms by tracking the turnaround time and waiting time for each process and printing the average waiting time, average turnaround time and throughput after all processes have completed.

- The turnaround time for a process is the difference between a job's submission and completion times. The average turnaround time reports the average for the set of processes that were scheduled.

- The waiting time for a process reflects the total amount of time spent by that process in the ready queue. The average waiting time reports the average for the set of processes that were scheduled.

- The throughput for a scheduling algorithm measures the total number of tasks processes per unit time.

2. Task Requirements

1. Your program must accept two arguments from the command line. The first argument is the name of the file containing the processes (for example: processinfo.csv). This file is comma-separated with four columns (process ID, arrival time, burst time, priority) with each row for an individual process. You can assume that this file will have a maximum of 10 processes. The second argument is the time quantum for Round Robin scheduling.

2. Your program (sch.c or sch.py) should be able to perform FCFS, Priority Scheduling, and Round Robin scheduling correctly given a set of processes and arrival time and burst time and priority for each process.

3. Implement FCFS and print a Gantt chart showing the execution sequence using the format shown below (align the number in the columns). Report waiting time and turnaround time for each process. Also, report the average values for waiting time and turnaround time, and the overall throughput for all the processes.

1. Repeat item 3 above for Priority Scheduling.

2. Repeat item 3 above for RR using the specified quantum.

Note: The processes in the file (first command line argument file) may be specified in such a way that it may result in some IDLE time when there are no processes ready to be executed. During the IDLE time the CPU has no processes to execute and waits for the next process to appear in the ready queue. Your Gantt chart should include these IDLE times. Also, the first process need not start at time 0. At time 0, there can be IDLE time in the Gantt chart till the first process arrives later on in time.

3. Example Input

Input File: processinfo.csv

```
ProcessID,Arrival Time,Burst Time,Priority
3,0,2,2
2,0,5,4
1,10,8,1
4,10,5,3
```

Note: The first line in the processinfo.csv file has headers Process ID, Arrival Time, Burst Time, Priority.

4. Example Output

```
<system_name>:<folder_path>$ python3 sch.py processinfo.csv 4

----------------- FCFS -----------------
Process ID | Waiting Time | Turnaround Time
    1      |      0       |        8
    2      |      0       |        5
    3      |      5       |        7
    4      |      8       |        13

Gantt Chart is:
[  0   ]--  2   --[  5   ]
[  5   ]--  3   --[  7   ]
[  7   ]-- IDLE --[  10  ]
[  10  ]--  1   --[  18  ]
[  18  ]--  4   --[  23  ]

Average Waiting Time:  3.25
Average Turnaround Time:  8.25
Throughput:  0.173913043478


------------------ PS ------------------
  Process ID |  Waiting Time  |  Turnaround Time
```

```
    1    |    0    |    8
    2    |    2    |    7
    3    |    0    |    2
    4    |    8    |    13
Gantt Chart is:
[    0    ]-- 3    --[ 2    ]
[    2    ]-- 2    --[ 7    ]
[    7    ]-- IDLE--[ 10   ]
[    10   ]-- 1    --[ 18   ]
[    18   ]-- 4    --[ 23   ]
Average Waiting Time: 2.5
Average Turnaround Time: 7.5
Throughput: 0.17391304347826086
----------------- Round Robin -----------------
Process ID | Waiting Time | Turnaround Time
    1      |      4       |        12
    2      |      2       |        7
    3      |      4       |        6
    4      |      8       |        13

Gantt Chart is:
[  0   ]--  2   --[  4   ]
[  4   ]--  3   --[  6   ]
[  6   ]--  2   --[  7   ]
[  7   ]-- IDLE --[  10  ]
[  10  ]--  1   --[  14  ]
[  14  ]--  4   --[  18  ]
[  18  ]--  1   --[  22  ]
[  22  ]--  4   --[  23  ]

Average Waiting Time:  4.5
Average Turnaround Time:  9.5
Throughput:  0.173913043478
```

Note: For the output format, it can be slightly different as long as you respect the same columns and rows and it must be easily readable.

The format for command line arguments for grading will be:

```
> ./schedule <filename> <time quantum>
```

(or)

```
> python3 sch.py <filename> <time quantum>
```

The filename is the name of the file containing the information about all the processes.

5. What to Submit

Use the CS370 *Canvas* to submit a single .zip or .tar file that contains:

- All .c or .py files with descriptive comments within,
- a Makefile that performs both a *make build* as well as a *make clean* if you are submitting .c files

- a README.txt file containing a description of each file and any information you feel the grader needs to grade your program, and answers for the 5 questions

For this and all other assignments, ensure that you have submitted a valid .zip/.tar file. After submitting your file, you can download it and examine to make sure it is indeed a valid zip/tar file, by trying to extract it.

Filename Convention: The archive file must be called: <FirstName>-<LastName>-HW4.<tar/zip>. E.g. if you are John Doe and submitting for assignment 4, then the tar file should be named John-Doe-HW4.tar

6. Grading

The assignments much compile and function correctly on machines in the CSB-120 Lab. Assignments that work on your laptop on your particular flavour of Linux/Mac OS X but not on the Lab machines are considered unacceptable.

The grading will be done on a 100-point scale.

The points are broken up as follows:

Objective		**Points**
Gantt Chart	FCFS	12.5 points
	PS	22.5 points
	Round Robin	32.5 points
Throughput	FCFS	2.5 points
	PS	2.5 points
	Round Robin	2.5 points
Waiting time and Average Waiting time	FCFS	2.0 points (W.T.) and 0.5 points(A.W.T.)
	PS	2.0 points (W.T.) and 0.5 points(A.W.T.)
	Round Robin	2.0 points (W.T.) and 0.5 points(A.W.T.)
Turnaround Time and Average Turnaround time	FCFS	2.0 points (T.A.T.) and 0.5 points(A.T.A.T.)
	PS	2.0 points (T.A.T.) and 0.5 points(A.T.A.T.)
	Round Robin	2.0 points (T.A.T.) and 0.5 points(A.T.A.T.)
Compilation with no warnings or errors		2 points
Suitable documentation of code in code files and README		3 points
Questions in the README file		5 points

Questions: (To be answered in README file)

1. Is the Shortest Remaining Time First a non-preemptive Algorithm?
2. What are the 5 different states a process can be in scheduling (Look into process state diagram)?

3. Shortest Job First is like Priority Scheduling with the priority based on _____ of the process?
4. _____ effect is the primary disadvantage of First Come First Serve Scheduling algorithm.
5. How does Multi Level Feedback queue prevent starvation of processes that waits too long in lower priority queue?

You are required to **work alone** on this assignment.

7. Late Policy

Click here for the class policy on submitting late assignments

Revisions: Any revisions in the assignment will be noted below.