CS39002 Operating Systems Laboratory
Spring 2025

Lab Assignment: 3
Date of submission: 22–Jan –2025

Performance Comparison of Some Scheduling Algorithms

In this assignment, you compare the performance of FCFS (First-Come-First-Serve) scheduling with that of RR (Round-Robin) scheduling, and also the effect of the time quantum q on the performance of RR scheduling. You deal with the case of a single CPU. FCFS scheduling can be considered as a special case of RR scheduling with $q = \infty$, so you only work on RR scheduling, and specify q as a parameter. You do not deal with real processes, but simulate the behavior of the CPU on processes with random behavior.

The details of the processes are stored in a text file *proc.txt*. The file starts with the number n of processes that you are going to deal with. Each of the remaining n lines in the file stores the following information about a process. The lines may appear in any sequence (not necessarily sorted by IDs or arrival times).

ID ARRIVAL_TIME CPU_1 IO_1 CPU_2 IO_2 . . . CPU_b -1

Each process is given a unique integer ID (you may think of this as the PID of the process). This is followed by the time when the process arrives (a non-negative integer). This is followed by a sequence of alternate CPU- and IO-burst times (positive integers). The list ends with –1 as an IO-burst time indicating the end of the process. Assume that all times are in milliseconds. In reality, the burst times are not known beforehand, but you deal with the performance of the scheduling algorithms on the given processes, so assume that these burst times are exact. You are not required to implement SJF scheduling, so whether or not you can estimate burst times is not a concern to this assignment.

A process-generator program is supplied to you as *genproc.c*. Compile and run the code with n (number of processes) as an optional command-line argument. Without this argument, n = 100 is taken. This generates data for n processes with the following randomly chosen parameters.

Process type	Probability	CPU burst time	IO burst time	# CPU bursts	# IO burst
IO-bound	90%	1 – 15	50 – 200	4 – 10	3 – 9
CPU-bound	10%	100 – 300	50 – 200	3 – 7	2 – 6

Each process is assumed to start and end with CPU bursts, so the number of IO bursts is one less than the number of CPU bursts. You may use these statistics to determine your array sizes.

Part 1: Read the input file

Define a structure to store the information about the processes. Each process requires storage for the ID, the arrival time, the number of bursts, and the burst times. In addition, you store additional information (like state of the process, which burst it is in, and so on) as needed by the simulation. It is your choice whatever you need to store in each process record. Read *proc.txt*, and populate an array of process-info structures. The queues to be used in the simulation will consist only of indices in this array (and nothing else). In essence, each entry in the process-info array *behaves* like the PCB of a process.

Part 2: Data structure for the Ready Queue

For both FCFS and RR scheduling, the ready queue is implemented as a FIFO queue. Design a data structure to perform the init, front, enqueue, dequeue (and other operations) on a FIFO queue. Each element in the queue will be an index identifying the process in the process-info table. So it will be a queue of integers. Make your own queue implementation.

Part 3: Handling the events [Discrete-event system simulation]

Your simulation neither creates any of the *n* processes nor runs the process on any CPU. On the contrary, your program assumes that the processes behave as stored in the process-info array, and schedules the processes on an imaginary CPU. In order to do so, you need to handle events *chronologically* starting at time 0. An *event* in the simulation is an incident that occurs at a particular time and that changes the configuration of the simulated system. In particular, you need to handle the following events.

- Arrival of a new process
- End of a process
- End of the use of CPU (after timeout or CPU-burst end)
- Rearrival of a process after IO completion

These events must be handled in a non-decreasing order of the times when they happen. A min-priority queue is used for that purpose, and is called the *event queue*. Like the ready queue, the elements of the queue are integers which are indices in the process-info table. At the beginning of the simulation, only the *n* arrival times of the processes are known. The event queue should be initialized by these *n* arrivals. Later, when the simulation proceeds, the exact times of the other types of events will be calculated, and inserted in the event queue. The first ordering parameter for the even queue is the times of occurrences of the events. For multiple events happening at the same time (like two processes complete IO and a new process arrives at the same time), some tie-breaking policies are to be used. In respect of the joining of a process to the ready queue, the following ordering is to be maintained:

Arrival (for the first time or after IO completion) < CPU timeout

Ties are possible even with this ordering. Give precedence to smaller IDs to break the ties. For example, suppose that a process with ID j arrives at the same time when a process with ID i completes an IO burst. If we have $i < j$, then the IO-completion event will appear earlier in the event queue than the new-arrival event.

In order to handle the next event, extract the first element from the event queue. Set the current time to the time of the event. Depending on the type of the event, do the following.

Arrival of a process for the first time or after IO completion: Put the process at the back of the ready queue.

CPU burst ends: Check whether this was the last CPU burst of the process. If so, print that the process exits, and also the following performance measures pertaining to that process.

- Turnaround time
- Turnaround time as a percentage of running time (total CPU + IO burst times)
- Wait time (turnaround time – running time)

Otherwise (that is, the process has more bursts to do), the next burst of the process will be an IO burst. Add the time of that IO burst to the current time. Insert the (re)arrival event of that process after IO completion, to the event queue.

CPU timeout: The current CPU burst is preempted, that is, more time is needed to complete the current CPU burst, so insert the process at the back of the ready queue.

After handling each event, check whether the CPU is free. If so, and if the ready queue is not empty, schedule the process at the front of the ready queue for the next time quantum q or for the rest of the next CPU burst time (whichever is smaller).

Eventually, all the n processes completes all the bursts, and the event queue becomes empty. Stop the simulation at that point.

Make your own array-based implementation of the event queue as a (binary) min-heap.

Part 4: Performance figures

Print the per-process performance figures as explained in Part 3. In addition, print the following aggregate information after the simulation stops.

- Average wait time of a process
- Total turnaround time (simulation end time – simulation start time (0))
- Total idle time of the CPU in the simulation interval (total turnaround time)
- Percentage utilization of the CPU during the simulation interval

Part 5: main() function

In your main() function, call the scheduler for $q = \infty$ (a large integer like 10^9) to simulate FCFS scheduling. Then make two other calls of the scheduler to simulate RR scheduling with $q = 10$ and $q = 5$.

Part 6: Verbose/Concise output

The default behavior of your code would be to print only the per-process and aggregate performance figures. Enable verbose output by a compile-time flag VERBOSE. In the verbose mode, print all individual events (arrival/rearrival, departure, CPU end, scheduling decision). The printing format will be specified at the end.

Makefile

You may use the following makefile.

```
compile: schedule.c
	gcc -Wall -o schedule schedule.c

run: compile
	./schedule

vcompile: schedule.c
	gcc -Wall -o schedule -DVERBOSE schedule.c

vrun: vcompile
	./schedule

db: genproc.c
	gcc -Wall -o genproc genproc.c

clean:
	-rm -f genproc schedule proc.txt
```

Submit a single file *schedule.c*.

Sample Input

```
20
    1       0   11  115    8  144   10  111    3  109   12   76    5  170    9  -1
    2      91   14  129    8   84    2  131    8  186   15   67    4  -1
    3     493   13  138    1   60    1  134    9   64    9  176    5   50    6  144    5  198   14  -1
    4     796    4   91   12  134   13   51    7  118    3   72   11   80    5  174    3  190   13  -1
    5     978  170   90  227   72  165  179  207   98  113  141  140  -1
    6    1116   14   53    4   65    7   80    8  108   12  196   12  145    2  136    2  185    7  181    1  -1
    7    1347   15   69    5   86    6  160    4  139   11  134   10  -1
    8    1683   12  185    5  100   10  147    6  162    3  -1
    9    1774   14  166    8   75   11   75    8   69   15   52    2  128   10  -1
   10    2172   11  157   13   92   14  155   14  118    9  136   10  130    2  -1
   11    2249   12  164    3  105   15  119   15   98   14  102    7   57   12  184   15   89    7  -1
   12    2674    5   72    8  124   15  175    7  135   11  137    3   81    7  161   14   86   13  -1
   13    2884    5  106    2   97   10  186    8  -1
   14    2998    5  130    1  150    3  190    5   51    1   68   15   56   14  186   14  185   12  -1
   15    3222   15  100   11   85    6   92   15   79    2  124    9   70    8  124    2   53    1   92   14  -1
   16    3587    3  156    7  150    7  147   11  127    3   77   11  102   14  -1
   17    3724    6  161   11  186    9   53   15  149    3  -1
   18    3761   14   96   10  133    8   75    8  136    1  173    5   64   13  188    8  107   15  -1
   19    3958   14  186    4  145   14   92    9  195    9   81   10  154    7  -1
   20    4335  280  116  221  142  147   70  175   81  134   69  190  111  237  -1
```

Sample Output (without the VERBOSE flag)

```
**** FCFS Scheduling ****
739       : Process      2 exits. Turnaround time =  648 (100%), Wait time = 0
783       : Process      1 exits. Turnaround time =  783 (100%), Wait time = 0
1752      : Process      3 exits. Turnaround time = 1259 (123%), Wait time = 232
2363      : Process      4 exits. Turnaround time = 1567 (160%), Wait time = 586
2588      : Process      5 exits. Turnaround time = 1610 (100%), Wait time = 8
2598      : Process      7 exits. Turnaround time = 1251 (196%), Wait time = 612
2760      : Process      9 exits. Turnaround time =  986 (156%), Wait time = 353
2771      : Process      8 exits. Turnaround time = 1088 (173%), Wait time = 458
2994      : Process      6 exits. Turnaround time = 1878 (154%), Wait time = 660
3194      : Process     10 exits. Turnaround time = 1022 (119%), Wait time = 161
3304      : Process     13 exits. Turnaround time =  420 (101%), Wait time = 6
3464      : Process     11 exits. Turnaround time = 1215 (119%), Wait time = 197
3750      : Process     12 exits. Turnaround time = 1076 (102%), Wait time = 22
4097      : Process     14 exits. Turnaround time = 1099 (101%), Wait time = 13
4134      : Process     15 exits. Turnaround time =  912 (101%), Wait time = 10
4329      : Process     17 exits. Turnaround time =  605 (102%), Wait time = 12
4629      : Process     16 exits. Turnaround time = 1042 (128%), Wait time = 227
5091      : Process     18 exits. Turnaround time = 1330 (126%), Wait time = 276
5248      : Process     19 exits. Turnaround time = 1290 (140%), Wait time = 370
6308      : Process     20 exits. Turnaround time = 1973 (100%), Wait time = 0

Average wait time = 210.15
Total turnaround time = 6308
CPU idle time = 2752
CPU utilization = 56.37%

**** RR Scheduling with q = 10 ****
739       : Process      2 exits. Turnaround time =  648 (100%), Wait time = 0
783       : Process      1 exits. Turnaround time =  783 (100%), Wait time = 0
1548      : Process      3 exits. Turnaround time = 1055 (103%), Wait time = 28
1838      : Process      4 exits. Turnaround time = 1042 (106%), Wait time = 61
2040      : Process      7 exits. Turnaround time =  693 (108%), Wait time = 54
2336      : Process      8 exits. Turnaround time =  653 (104%), Wait time = 23
2426      : Process      6 exits. Turnaround time = 1310 (108%), Wait time = 92
2461      : Process      9 exits. Turnaround time =  687 (109%), Wait time = 54
2918      : Process      5 exits. Turnaround time = 1940 (121%), Wait time = 338
3077      : Process     10 exits. Turnaround time =  905 (105%), Wait time = 44
3317      : Process     13 exits. Turnaround time =  433 (105%), Wait time = 19
3336      : Process     11 exits. Turnaround time = 1087 (107%), Wait time = 69
3765      : Process     12 exits. Turnaround time = 1091 (104%), Wait time = 37
4132      : Process     14 exits. Turnaround time = 1134 (104%), Wait time = 48
4146      : Process     15 exits. Turnaround time =  924 (102%), Wait time = 22
4358      : Process     17 exits. Turnaround time =  634 (107%), Wait time = 41
4441      : Process     16 exits. Turnaround time =  854 (105%), Wait time = 39
4879      : Process     18 exits. Turnaround time = 1118 (106%), Wait time = 64
4906      : Process     19 exits. Turnaround time =  948 (103%), Wait time = 28
6383      : Process     20 exits. Turnaround time = 2048 (104%), Wait time = 75

Average wait time = 56.80
Total turnaround time = 6383
CPU idle time = 2827
CPU utilization = 55.71%

**** RR Scheduling with q = 5 ****
739       : Process      2 exits. Turnaround time =  648 (100%), Wait time = 0
783       : Process      1 exits. Turnaround time =  783 (100%), Wait time = 0
1543      : Process      3 exits. Turnaround time = 1050 (102%), Wait time = 23
1828      : Process      4 exits. Turnaround time = 1032 (105%), Wait time = 51
2037      : Process      7 exits. Turnaround time =  690 (108%), Wait time = 51
2334      : Process      8 exits. Turnaround time =  651 (103%), Wait time = 21
2419      : Process      6 exits. Turnaround time = 1303 (107%), Wait time = 85
2444      : Process      9 exits. Turnaround time =  670 (106%), Wait time = 37
2916      : Process      5 exits. Turnaround time = 1938 (121%), Wait time = 336
3065      : Process     10 exits. Turnaround time =  893 (104%), Wait time = 32
3307      : Process     13 exits. Turnaround time =  423 (102%), Wait time = 9
3314      : Process     11 exits. Turnaround time = 1065 (105%), Wait time = 47
3780      : Process     12 exits. Turnaround time = 1106 (105%), Wait time = 52
4092      : Process     14 exits. Turnaround time = 1094 (101%), Wait time = 8
4143      : Process     15 exits. Turnaround time =  921 (102%), Wait time = 19
4331      : Process     17 exits. Turnaround time =  607 (102%), Wait time = 14
4429      : Process     16 exits. Turnaround time =  842 (103%), Wait time = 27
4866      : Process     18 exits. Turnaround time = 1105 (105%), Wait time = 51
4908      : Process     19 exits. Turnaround time =  950 (103%), Wait time = 30
6380      : Process     20 exits. Turnaround time = 2045 (104%), Wait time = 72

Average wait time = 48.25
Total turnaround time = 6380
CPU idle time = 2824
CPU utilization = 55.74%
```

Sample Output (with the VERBOSE flag)

```
. . .

**** RR Scheduling with q = 10 ****
0         : Starting
0         : Process 1 joins ready queue upon arrival
0         : Process 1 is scheduled to run for time 10
10        : Process 1 joins ready queue after timeout
10        : Process 1 is scheduled to run for time 1
11        : CPU goes idle
91        : Process 2 joins ready queue upon arrival
91        : Process 2 is scheduled to run for time 10
101       : Process 2 joins ready queue after timeout
101       : Process 2 is scheduled to run for time 4
105       : CPU goes idle
126       : Process 1 joins ready queue after IO completion
126       : Process 1 is scheduled to run for time 8
134       : CPU goes idle
234       : Process 2 joins ready queue after IO completion
234       : Process 2 is scheduled to run for time 8
242       : CPU goes idle
278       : Process 1 joins ready queue after IO completion
278       : Process 1 is scheduled to run for time 10
288       : CPU goes idle
326       : Process 2 joins ready queue after IO completion
326       : Process 2 is scheduled to run for time 2
328       : CPU goes idle
399       : Process 1 joins ready queue after IO completion
399       : Process 1 is scheduled to run for time 3
402       : CPU goes idle
459       : Process 2 joins ready queue after IO completion
459       : Process 2 is scheduled to run for time 8
467       : CPU goes idle
493       : Process 3 joins ready queue upon arrival
493       : Process 3 is scheduled to run for time 10
503       : Process 3 joins ready queue after timeout
503       : Process 3 is scheduled to run for time 3
506       : CPU goes idle
511       : Process 1 joins ready queue after IO completion
511       : Process 1 is scheduled to run for time 10
521       : Process 1 joins ready queue after timeout
521       : Process 1 is scheduled to run for time 2
523       : CPU goes idle
599       : Process 1 joins ready queue after IO completion
599       : Process 1 is scheduled to run for time 5
604       : CPU goes idle
644       : Process 3 joins ready queue after IO completion
644       : Process 3 is scheduled to run for time 1
645       : CPU goes idle
653       : Process 2 joins ready queue after IO completion
653       : Process 2 is scheduled to run for time 10
663       : Process 2 joins ready queue after timeout
663       : Process 2 is scheduled to run for time 5
668       : CPU goes idle
705       : Process 3 joins ready queue after IO completion
705       : Process 3 is scheduled to run for time 1
706       : CPU goes idle
735       : Process 2 joins ready queue after IO completion
735       : Process 2 is scheduled to run for time 4
739       : Process      2 exits. Turnaround time =  648 (100%), Wait time = 0
739       : CPU goes idle
774       : Process 1 joins ready queue after IO completion
774       : Process 1 is scheduled to run for time 9
783       : Process      1 exits. Turnaround time =  783 (100%), Wait time = 0
783       : CPU goes idle

. . .

6296      : Process 20 is scheduled to run for time 10
6306      : Process 20 joins ready queue after timeout
6306      : Process 20 is scheduled to run for time 10
6316      : Process 20 joins ready queue after timeout
6316      : Process 20 is scheduled to run for time 10
6326      : Process 20 joins ready queue after timeout
6326      : Process 20 is scheduled to run for time 10
6336      : Process 20 joins ready queue after timeout
6336      : Process 20 is scheduled to run for time 10
6346      : Process 20 joins ready queue after timeout
6346      : Process 20 is scheduled to run for time 10
6356      : Process 20 joins ready queue after timeout
6356      : Process 20 is scheduled to run for time 10
6366      : Process 20 joins ready queue after timeout
6366      : Process 20 is scheduled to run for time 10
6376      : Process 20 joins ready queue after timeout
6376      : Process 20 is scheduled to run for time 7
6383      : Process     20 exits. Turnaround time = 2048 (104%), Wait time = 75
6383      : CPU goes idle

Average wait time = 56.80
Total turnaround time = 6383
CPU idle time = 2827
CPU utilization = 55.71%

. . .
```