Alexandria University
Faculty of Engineering
Comp. & Comm. Engineering
CC373: Operating Systems

جامعة الاسكندرية
كلية الهندسة
برنامج هندسة الحاسب والاتصالات
مادة نظم التشغيل

Lab4
CPU Scheduling

Introduction:

You are asked to implement one program in C/C++ to analyze, and visualize the following CPU Scheduling algorithms. You are encouraged to use STL so as to reduce the implementation effort (you do not need to implement queues or priority queues yourself):

1. FCFS (First Come First Serve)
2. RR (Round Robin)
3. SPN (Shortest Process Next)
4. SRT (Shortest Remaining Time)
5. HRRN (Highest Response Ratio Next)
6. FB-1, (Feedback where all queues have q=1)
7. FB-2i, (Feedback where q= 2^i)
8. Aging

Getting Input from User:

The input to your program will be sent through `stdin`. Your output should be sent to `stdout`. To test your program, you are provided with a set of testcases showing the format of the input. You do not have to type the input yourself. You can simply use input redirection (e.g., `./lab4 < 01a-input.txt`), or pipe the input to your binary (e.g., `cat 01a-input.txt | ./lab4`).

Input Format:

Line1. “trace” or “stats”.
“trace” is used to ask your program to visualize the processes switching over the CPU – just like the Figure 9.5 in your textbook (or slide 30 in the class presentation). You can see `01a-output.txt` as an example.
“stats” is used to ask your program to write some statistics on the scheduled processes – just like Table 9.5 in your textbook (or slide 31 in the class presentation). You can see `01b-output.txt` as an example.
used to visualize the processes

Line2. a comma-separated list telling which CPU scheduling policies to be analyzed/visualized along with their parameters, if applicable. Each algorithm is represented by a number as listed in the introduction section and as shown in the attached testcases.
Round Robin has a parameter specifying the quantum to be used. Therefore, a policy entered as 2-4 means Round Robin with quantum = 4.

Line3. An integer specifying the last instant to be used in your simulation and to be shown on the timeline. Check the attached testcases.

Line4. An integer specifying the number of processes to be simulated. None of the processes is making a blocking call.

Line5. Start of description of processes. Each process is to be described on a separate line. For algorithms 1 through 7, each process is described using a comma-separated list specifying:
- One character specifying a process name
- Arrival Time
- Service Time

Output Format:

As mentioned earlier, your output will be sent to `stdout`. You must provide a Makefile through which your program is to be compiled to produce a binary called `lab4` in the same directory as the source code. Your program will be automatically graded. So, adhering to the output format is a must, else your program will fail the used testcases during grading. Check the files representing the output in the provided testcases for the strict format.

When visualizing the processes, you will use an asterisk "*" to show that the process is running in this period. You will use a period "`.`" To specify that the process was in the ready list at this time period.

In addition to any test case you write to test your code to make sure it takes care of boundary conditions, race conditions, …, you can use the provided testcases to verify that your code produces the right output format. You can use the following commands:

```
cat 01a-input.txt | ./lab4 | diff 01a-output.txt -
cat 01b-input.txt | ./lab4 | diff 01b-output.txt -
...
```

These commands show the differences (hence the command `diff`) between your program's output and the expected output. In case of discrepancies, your program fails the testcase.

Aging Scheduling Policy:

You will be provided details soon.

Deliverables:

You will be provided details soon.

Note:

This is a team project. The team size is at most 2. One submission will be made for the project.