Operating Systems

Department of Computer Science
Aristotle University of Thessaloniki

2nd Assignment Fall
Semester 2022

The aim of the assignment is to familiarise you with the routing algorithms of interconnection.

For this purpose, you are asked to develop *C-language* code that implements the FCFS, SJF (without preemption), SRTF (with preemption) and Round Robin (for any quantum value) routing algorithms.

In the *code* directory that accompanies the pronunciation, you will find the C programming language files on which you should base your deliverables. These files are:

- fcfs.c: File where the FCFS algorithm should be implemented
- sjf.c: File where the SJF algorithm should be implemented
- srtf.c: File where the SRTF algorithm should be implemented
- rr.c: File where the Round Robin algorithm should be implemented
- Makefile: make file for compiling the code and performing checks.

In each of the C code files a struct is implemented, in which information about each process is stored. You can add additional properties to this struct, but you cannot remove properties that already exist that relate to the process ID, arrival time, and CPU bust time.

Then, at the beginning of the main function, there is a piece of code implemented which reads some input data from the standard input (stdin) and processes it, populating a table with struct process elements. You should not change this piece of code.

You can add additional functions, variables and structs according to your needs.

The work requires running on a Linux environment and using the C language development tools available on it (gcc compiler and au- tomake).

The compile of the code for each algorithm can be done using the tool make, giving the name of the algorithm (fcfs, sjf, srt, rr) as a parameter, e.g.:

```
make fcfs
```

or in total for all algorithms, just running:

```
make
```

The result is that the corresponding executable files will be created, which can be executed by giving as their standard input files like *tests/INPUT01*, e.g:

```
cat tests/INPUT01 | ./fcfs
```

The input files should be plain text files and follow the following format:

- The first line contains an integer, which refers to the number of interpretations described below.
- The second line contains an integer, which refers to the quantum that the algorithms may use. The quantum is always defined, for all algorithms, but will obviously not be used by algorithms in which it plays no role.
- The remaining lines (as many as the integer in the first line of the file) contain three integers, representing the process ID, its arrival time, and its CPU storm time.

You can make the following assumptions:

- Input files will always be in the correct format.
- The processes will be given in the order of their arrival.
- There will be no gaps in the routing of processes, such as all other processes finishing running at a certain time, but no next process has yet arrived.

The code that you fill in the C code files should at the end of its execution print in the standard output, on each line, the ID of the process running at each time (starting at time 0). For the processes defined in *tests/INPUT01,* the output that should be generated will be the one in the corresponding files *tests/fcfs01*, *tests/sjf01*, *tests/srtf01* and *tests/rr01*, e.g:

```
$ cat tests/INPUT01 | ./fcfs
1
1
1
2
2
3
```

You can check whether the output generated by your executables corresponds to them by running for each algorithm a command such as

```
make test-fcfs
```

or for all algorithms in total:

```
make test
```

The minimum requirement for the task is that for the one test case given (*tests/INPUT01*) by running the make test command, nothing is printed to the terminal (so there is a mapping of the output of the programs to that found in the *tests/fcfs01* files etc.

In case there are errors and the programs do not generate the output they should, there will be messages like the example below:

```
$ make
$ make test
Files tests/fcfs01 and /dev/fd/63 differ
Files tests/sjf01 and /dev/fd/63 differ Files
tests/srtf01 and /dev/fd/63 differ Files
tests/rr01 and /dev/fd/63 differ
```

You can add new test cases by creating files named *INPUTXX*, *fcfsXX*, *sjfXX*, *srtfXX* and *rrXX* in the *tests* folder, where XX is an integer with two digits (e.g. INPUT02 etc.) to better check the correct implementation of the algorithms.

You should *not* accompany your code with any external text, commenting should be done solely on the code.

To submit your work, you must run the command:

```
make tar
```

which will generate a *scheduling.tar.gz* file, which you will submit via the e-learning platform.

There is also the clean rule, which deletes the executable files that have been created and the scheduling.tar.gz file, if it exists:

```
make clean
```

The *Makefile* file may not be changed in any way.

The work is individual. Any form of copying is prohibited.

If you have questions about the project, you can submit them via our elearning platform in the corresponding discussion forum.