Analysis 4
Operating Systems Assignment

In this assignment, we will simulate the behaviour of the short-term scheduler.

Modern operating systems run multiple tasks concurrently. Assuming only a single CPU is available, multitasking is achievable by sharing CPU time. However, to make processes responsive to user requests, each process can use the CPU for only a limited time, named a **quantum**. At the end of the quantum, the OS will clear the CPU and the process will wait for its next quantum in a queue named the **ready queue** (see the course material for more information). Managing the ready queue is the task of the short-term scheduler. One of the algorithms used by the short-term scheduler is the Round Robin algorithm. In Round Robin, each process uses the CPU for one quantum and then is added to the end of the ready queue (for this assignment there will be no priority queues). This algorithm is applied until all the processes are terminated. When one process has terminated its execution time it is permanently removed from the scheduling.

In the simulation, you will have several "fake processes" to schedule, and your script needs to decide when one of the "fake processes" needs to run. Each "execution" time for each quantum will be represented only from **a printout**. After the printing process is done, the simulation will continue with the next running process.

Here is an example of behaviour:

Assume the following data about the sample processes are given.

Process Name	Start Time*	Execution Time**
P1	0	4
P2	1	2
P3	3	1

*Start time shows the time of arrival of each process.

**Execution time shows the number of quanta that the process needs to finish its job. Each quantum duration is one unit of time (they will all be CPU bursts).

Each “process” after accessing the CPU prints a message on the screen. The operating system prints a message when a process terminates. Therefore, the execution of the processes shown above will create the following output.

P1 is using the CPU

P2 is using the CPU

P1 is using the CPU

P2 is using the CPU

Process P2 terminated

P3 is using the CPU

Process P3 terminated

P1 is using the CPU

P1 is using the CPU

Process P1 terminated

Implementation details

Write a Linux shell script (bash) to simulate the behaviour of the short-term scheduler using a round-robin algorithm as shown above.

The script must load a process file and simulate the scheduler behaviour according to the round-robin algorithm for the execution of each process.

INPUT: Your script should read the process data from an input file where each row includes comma-separated data about one process in each row. An example of the data will follow below.

Ex. of execution:

simulator.sh *-file* maybeexistingmaybenotexistingfile.csv

You must handle errors in input (missing files, wrong access rights, not formatted properly etc…). It is acceptable to terminate execution in case of errors, it is not acceptable to let it crash or misbehave, errors must be handled not just suppressed.

Example of a data file:

P1,0,4

P2,1,3

In the example, P1 is the name of the first process and P2 of the second. The first number stands for when it is enqueued and the second is the duration of its execution in quantum.

Mini example: *P1, 0, 4 → process1 will be executed at time 0, at time 1 will be reactivated, at time 2 will be reactivated, and at time 3 will be reactivated for the last time as no other process is running.*

If 2 or more Programs arrive at the same time, they will be scheduled in the order of entry.

To simplify your work, we assume that each program is running as represented by its printout, so you do not need to represent a context switch or memory swap. So P1 and P2 are representatives of the processes that are running but you will just be using the printout (no need to invoke any program or command, it is only a simulation, and every printout will represent the scheduling).

After each complete run of a process, it should print "*Process PP terminated*" where "PP" is the name of the process that terminated.

If there are no programs scheduled for some quantum, the simulation must run the program "idle" in substitution.

Hint: You may use a temporary file to implement the queue. There are many ways to do it, we suggest using what we learned in the course.

The script must not ask for root/administrative permissions/rights.

The script must never ask to install or need external resources that are not already installed in the standard distribution we are using.

THE SCRIPT MUST BE ABLE TO BE EXECUTED FROM ANY PLACE IN THE HOME DIR or SUB DIR OF THE CURRENT USER.

Testing will be done by creating different versions of input files and running them from different directories in the user's home. The file will be placed in any directory or subdirectory in the same partition where it is called.

Some Examples:

simulator.sh *-file* realfile.csv

simulator.sh *-file* filethatdesnotexist.csv

/usr/testuser/documents/# ../**simulator.sh** *-file* maybeexistingmaybenotexistingfile.csv

~**# simulator.sh** *-file* realfile.csv

Etc…

Deliverables:

Only the file of the script should be delivered.
The file should contain a comment (bash comment) on **the second line with the names and numbers of the delivering students.**
The name of the file will be **01234567_01234568_simulator.sh** or in the case of one student only **01234568_simulator.sh**

The assignment can be completed in groups of **UP TO TWO students**.

PAY ATTENTION TO:

- which interpreter is executing when you run the script.
- copy-paste/editing across operating systems can contaminate the file with “\r\n” (or a similar character as “^m”) that can hinder the execution. If that happens the assignment will be considered as failed. The file will be tested in the given VM, so please deliver from there to prevent accidental failures.

YOU MUST ADD COMMENTS THAT EXPLAIN YOUR DESIGN CHOICES.

As in the course description we will verify with oral checks any doubt or unclear code.

Delivery:

Deadline: 11:30 Friday 23 June.

Link to upload the assignment: TO COME