Report of Group Project 1 of Operating Systems

Emirhan Balcı Nº17048 Miguel Augusto Nº42650

University of Évora

1 Introduction

This report will be written in English due to it being a shared language in both partners of the group. The objective of this Group Work is to create a simulator of a Five-State Process Operating System. This Operating System should contain the States: NEW, READY, RUN, BLOCKED and EXIT. When the processes are in the states READY and BLOCKED they will be in saved in Queues of the type FIFO (First In, First Out).

2 project.c

The most important file of the project. Contains the main() and the functions that allow the program to run and give the desired Output. The state of each process is kept in an Array - whenever it needs to print its state, it accesses this Array. Each process can access to an Integer which states whether a process is running or not, since there can only be 1 process running at a time. It also contains a structure for each Process. Each Process has a variable for: - Its State, ranging from 0 to 5 (int state); - Its Id, being unique to that process, the first one being 0 (int id); - The time that each state can take (int timer); - The position that it is on its Array of states, mostly functions like the y in the int processes[x][y] that is given to us as an input (int index); - And a flag, that states whether the process has started, ended or is still running.

2.1 main()

The main contains the functions of the program and contains the input array. Its capacity is obtained via the function CountProcess(), which is then used to build the header (via the PrintHeader() function) and it's also used in the simulation (FCFS() function). It prints an "END!" once it's finished. This program also contains a Queue in order to organize the process order. Processes were chosen to be represented with a structure.

2.2 FCFS()

This is the function of our algorithm, "First Come, First Served". If a process needs to Run, its Id will be saved on a Queue, and the first process to Run will always be the first one that arrived in that Queue. This function returns

nothing, but instead, it prints the results of our program and executes everything accordingly. It initiates and saves each process in an Array (processArray) and saves their Id's in a Queue (processQueue), and once this Queue is Empty, the program stops and prints an "END!".

3 Queue

A Queue was used to organize the order of this processes in this program. It saves the values of the id of each process and uses that value to decide what process should be the one Running next. The file "queue.c" contains commands that allow us to create a Queue, Enqueuing and Dequeue id's, if it is full or empty, check the successor (the Id after a certain Id) and the Id that is next to Run. It also has a function called FrontAndDequeue(), which allows us to check both the Id on the Front and Dequeue it. Whenever a proccess is next to Run, it should be Dequeued and, whenever a value is Enqueued, it must wait on the READY state until it is ready to change to the RUNNING state.

4 Output

The following images are the examples of 2 of the outputs of the program. The first image is for the 1st Test (in order, given by the teachers) and the other 2 images are for the 4th Test (the last one). They end on Instant 33 and Instant 86, as desired.

Instants:	proc1	proc2	proc3
1:	NEW	NEW	
2:	RUN	READY	
3:	RUN	READY	NEW
4:	RUN	READY	READY
5:	BLOCK	RUN	READY
6:	READY	RUN	READY
7:	READY	RUN	READY
8:	READY	RUN	READY
9:	READY	BLOCK	RUN
10:	READY	BLOCK	RUN
11:	RUN	READY	BLOCK
12:	RUN	READY	READY
13:	BLOCK	RUN	READY
14:	BLOCK	RUN	READY
15:	READY	RUN	READY
16:	READY	RUN	READY
17:	READY	BLOCK	RUN
18:	READY	READY	RUN
19:	READY	READY	RUN
20:	READY	READY	RUN
21:	READY	READY	RUN
22:	READY	READY	RUN
23:	RUN	READY	BLOCK
24:	RUN	READY	READY
25:	RUN	READY	READY
26:	RUN	READY	READY
27:	EXIT	RUN	READY
28:		RUN	READY
29:		RUN	READY
30:		EXIT	RUN
31:			RUN
32:			RUN
33:			EXIT

END!

Instants:	proc1	proc2	proc3	proc4	proc5	proc6	proc7	proc8
1:	NEW	NEW						
2:	RUN	READY	NEW	NEW				
3:	BLOCK	RUN	READY	READY	NEW			
4:	BLOCK	RUN	READY	READY	READY	NEW	NEW	
5:	BLOCK	RUN	READY	READY	READY	READY	READY	NEW
6:	BLOCK	RUN	READY	READY	READY	READY	READY	READY
7:	READY	RUN	READY	READY	READY	READY	READY	READY
8:	READY	BLOCK	RUN	READY	READY	READY	READY	READY
9:	READY	BLOCK	RUN	READY	READY	READY	READY	READY
10:	READY	READY	BLOCK	RUN	READY	READY	READY	READY
11:	READY	READY	READY	RUN	READY	READY	READY	READY
12:	READY	READY	READY	BLOCK	RUN	READY	READY	READY
13:	READY	READY	READY	READY	RUN	READY	READY	READY
14:	READY	READY	READY	READY	RUN	READY	READY	READY
15:	READY	READY	READY	READY	RUN	READY	READY	READY
16:	READY	READY	READY	READY	RUN	READY	READY	READY
17:	READY	READY	READY	READY	BLOCK	RUN	READY	READY
18:	READY	READY	READY	READY	READY	RUN	READY	READY
19:	READY	READY	READY	READY	READY	RUN	READY	READY
20:	READY	READY	READY	READY	READY	RUN	READY	READY
21:	READY	READY	READY	READY	READY	BLOCK	RUN	READY
22:	READY	READY	READY	READY	READY	BLOCK	RUN	READY
23:	READY	READY	READY	READY	READY	READY	RUN	READY
24:	READY	READY	READY	READY	READY	READY	RUN	READY
25:	READY	READY	READY	READY	READY	READY	BLOCK	RUN
26:	READY	READY	READY	READY	READY	READY	BLOCK	RUN
27:	READY	READY	READY	READY	READY	READY	READY	RUN
28:	RUN	READY	READY	READY	READY	READY	READY	BLOCK
29:	RUN	READY	READY	READY	READY	READY	READY	BLOCK
30:	RUN	READY	READY	READY	READY	READY	READY	BLOCK
31:	RUN	READY	READY	READY	READY	READY	READY	BLOCK
32:	RUN	READY	READY	READY	READY	READY	READY	BLOCK
33:	BLOCK	RUN	READY	READY	READY	READY	READY	BLOCK
34:	BLOCK	RUN	READY	READY	READY	READY	READY	READY
35:	READY	BLOCK	RUN	READY	READY	READY	READY	READY
36:	READY	READY	RUN	READY	READY	READY	READY	READY
37:	READY	READY	RUN	READY	READY	READY	READY	READY
38:	READY	READY	RUN	READY	READY	READY	READY	READY
39:	READY	READY	RUN	READY	READY	READY	READY	READY
40:	READY	READY	RUN	READY	READY	READY	READY	READY
41:	READY	READY	BLOCK	RUN	READY	READY	READY	READY
42:	READY	READY	READY	RUN	READY	READY	READY	READY
43:	READY	READY	READY	RUN	READY	READY	READY	READY
44:	READY	READY	READY	RUN	READY	READY	READY	READY
45:	READY	READY	READY	RUN	READY	READY	READY	READY
46:	READY	READY	READY	BLOCK	RUN	READY	READY	READY
47:	READY	READY	READY	BLOCK	RUN	READY	READY	READY
48:	READY	READY	READY	READY	BLOCK	RUN	READY	READY
49:	READY	READY	READY	READY	READY	RUN	READY	READY
50:	READY	READY	READY	READY	READY	RUN	READY	READY
51:	READY	READY	READY	READY	READY	RUN	READY	READY
52:	READY	READY	READY	READY	READY	RUN	READY	READY
53:	READY	READY	READY	READY	READY	RUN	READY	READY
54:	READY	READY	READY	READY	READY	RUN	READY	READY
55:	READY	READY	READY	READY	READY	BLOCK	RUN	READY
56:	READY	READY	READY	READY	READY	BLOCK	RUN	READY
57:	READY	READY	READY	READY	READY	BLOCK	RUN	READY
58:	READY	READY	READY	READY	READY	BLOCK	RUN	READY
59:	READY	READY	READY	READY	READY	READY	BLOCK	RUN
60:	READY	READY	READY	READY	READY	READY	BLOCK	RUN
61:	READY	READY	READY	READY	READY	READY	BLOCK	RUN
62:	READY	READY	READY	READY	READY	READY	READY	RUN
63:	READY	READY	READY	READY	READY	READY	READY	RUN
64:	RUN	READY	READY	READY	READY	READY	READY	BLOCK
65:	EXIT	RUN	READY	READY	READY	READY	READY	READY
66:		RUN	READY	READY	READY	READY	READY	READY
67:		RUN	READY	READY	READY	READY	READY	READY
68:		EXIT	RUN	READY	READY	READY	READY	READY
69:			RUN	READY	READY	READY	READY	READY
70:			RUN	READY	READY	READY	READY	READY
71:			RUN	READY	READY	READY	READY	READY
72:			EXIT	RUN	READY	READY	READY	READY
73:				RUN	READY	READY	READY	READY
74:				RUN	READY	READY	READY	READY
75:				RUN	READY	READY	READY	READY
76:				EXIT	RUN	READY	READY	READY
77:					EXIT	RUN	READY	READY
78:						RUN	READY	READY
79:						EXIT	RUN	READY
80:							RUN	READY
81:							RUN	READY
82:							RUN	READY
83:							EXIT	RUN
84:								RUN
85:								RUN
86:								EXIT
END!								

5 Conclusion

The program runs according to its specifications and can complete each of the Tests given to it successfully. It may lack in simplicity but perform as expected.