Bilge Kaan Güneyli
2020400051

CMPE 322 PROJECT 2 REPORT
TO SCHEDULE OR NOT TO SCHEDULE?

IMPLEMENTATION DETAILS

The project is implemented using Ubuntu WSL on Windows.
There are two code files which are under the same directory:

main.c: contains the main algorithm

functions.c: contains two helper functions

All the necessary files are read by the code and they are assumed to be on the same directory as the main.c and functions.c file
You can run the program using the commands:

make

./scheduler

The algorithm works in such a manner:

1. Read the files and write the content in arrays
2. Hold all the processes in a file that holds them in the order of arrival and priority
3. Enter the loop
    a. Take the arrived processes in the ready queue
    b. Put the ready queue in the order of execution
    c. Check whether we have to do a context switch or not
    d. Execute the process
    e. Check whether the process has ended or not

DIFFICULTIES FACED IN IMPLEMENTATION

1. Keeping the ready queue always in order was difficult (Actually, there are some situations where the order is not correct but we were aware that this disorder occurs and it is harmless.).
2. The edge case which occurs when a gold process ends its last quantum and becomes a platinum was hard to implement. Checking a process's priority level after it gets preempted required separate checking. To be able to handle this case I had to change the whole implementation of the algorithm.
3. Controlling the updates when the process does not get preempted was challenging. Placement of the relative code was very important because certain indexes had to change at similar times.
4. Handling the round robin cases was challenging too. Since these cases included all the protocols (preemption, upgrading, special priority among relevant processes) implementation was so vulnerable and was easily affected by the other changes made for other cases.
5. Keeping the CPU burst count created some difficulties. Like the previous case, the code related to this was occurring in so many places and this created some overlaps and made this part open to be implemented correctly. I had to handle every possible situation separately.