Volkan Ozturk 2019400033

Project 2 Report

Implementation:

I implemented scheduler (preemptive priority scheduling) successfully. I satisfied the rules and conditions that are given in the project description. My implementation approach in step by step:

1) I first decided which data structures that I should use. Normally, the best data structure for implementing a scheduler would be priority queue/ queue with qsort etc.; but using those structures would make the project unnecessarily more complicated. Since we will always have maximum of 10 processes and same processes only comes maximum of one time, using regular array was more than enough. Thus, I decided to use array which stores structs of Process. I also defined the struct of Process, which has 9 integer fields.
2) Then, I implemented necessary helper functions which I would call from main; either directly or indirectly:
    a. *int type_comparator(int, int):* This function is used to compare two processes by type. The exact hierarchy given in the project description (Platinum > Silver = Gold) is obeyed. Two integer parameters are the indices of the processes in the main array named 'queue'.
    b. *int process_comparator(int, int):* This function is used to compare two processes according to the exact hierarchy given in the project description (Type > Priority > Time of Arrival > Alphabetical Order). This is the main comparison function to compare two different processes.
    c. *int preemption_detection(int):* This function is used to detect whether preemption is necessary. If, at the time of checking, a platinum or higher priority process is ready, preemption necessary. Preemption is not applicable to the Platinum type processes as required in the project description.
    d. *void execute(int):* This is the function to simulate the execution of the process whose index in the 'queue' array given as the parameter to the function. The execution rules in the project description obeyed exactly. The Platinum processes are executed immediately unless there exist a higher priority Platinum process(es) in the execution or in the queue and they are not subject to preemption. Silver and Gold type processes are executed in the Round Robin basis, whose time quantum are 80ms and 120ms respectively. They are also subject to priority-based preemption, i.e., preemption is applied if any higher priority or Platinum type process is ready at the time of checking.
    e. *Int choose_process():* This function is used to find the process which should be executed next. It iterates through the queue and returns the index of the process which should be executed according to the hierarchy in the project description.
3) In the main function, I first read the file 'definition.txt' and create the processes according to this file in the 'queue' array. Then, I find the first process to be executed at the time of 0. After the context switch, as required by the description, the main while loop starts with the execution of the first process. Then in continuous manner, until all

processes are completed, the code finds the next process and executes it; if next process is different than the old one also context switch is made.

Assumptions and Clarifications:

I followed all the assumptions given in the project description file. Some parts, the instructions that the processes are composed of, and the durations of the processes are hardcoded. All other assumptions in the project description file are also valid.

One and only assumption that is not given in the project description is the comparison method of the process ids. *Processes will be executed by the order of their names* was ambiguous, and I choose the alphabetical order as the comparison method.

Difficulties Met:

Implementing a real process scheduler is obviously very difficult task. But the assumptions given in the project description was very well balanced in terms of decreasing the difficulty while maximizing to learn the idea behind the schedulers in general. Thus, I find the project to be very informative and instructive to implement. After choosing the right and convenient data structures, I did not encounter with any major difficulties while implementing the scheduler. I also found this project to be easier than the first one since we do not have such edge-case issues as we had in the first project.