COMPE571 – Embedded Operating Systems – Programming Assignment 2

NOTE: You have to work in a UNIX environment (e.g., Ubuntu, MacOS, Raspbian OS, etc.) to implement this assignment. You have to use C programming.

NOTE: This is a group assignment. You are to form groups of 2 or 3.

Assignment Description

In this programming assignment, you are going to implement and compare different scheduling algorithms with respect to their performance. Your scheduling algorithm implementations will feature UNIX signals, checking and changing process statuses (start/suspend a process), and finally response time analyses.

You are given a sample C file that has the following:

- There is a function definition, named `myfunction()`, that performs prime factorization for all numbers up to the given input parameter (`param`). This function will be the main workload for the processes that will be created later, and the size of each workload will be defined by the `param` value each function is initialized with.
- The main function creates four processes using `fork()`. The process IDs of these newly created processes are stored in variables `pid1`, `pid2`, `pid3`, and `pid4`. Each child process calls `myfunction()`, with a predefined WORKLOAD size (`WORKLOAD1`, `WORKLOAD2`, `WORKLOAD3`, and `WORKLOAD4` which are defined at the top of the C file). Note that you will need to change these sizes later based on the experiments you will need to create (experiment case 3b).
- After creating each child process, the main process **pauses/suspends** each newly created process by sending the `SIGSTOP` signal to it via `kill()` function. Once this operation is done for all four child processes, all the newly created child processes are stopped, and ready for scheduling (see line 97 in the C file).
- After this point, the main process has to implement an algorithm to take care of scheduling of the child processes. The sample C file you are given includes a sample scheduling algorithm implemented. This sample scheduling algorithm is Round Robin with a time quantum of 1000 microseconds for each process. This sample implementation features calling `kill()` function with `SIGSTOP` and `SIGCONT` signals, in order to suspend and continue a process, respectively, `usleep()` function to insert a delay to ensure timing of scheduling decisions, and `waitpid()` function (with WNOHANG option) to see if a process finished and changed status in the system. You should analyze this sample scheduling implementation (lines 109 to 140) in detail so that you can use the necessary concepts (e.g., suspending/resuming a process, etc.) in your implementations accordingly.
  - Note that the scheduler we are implementing here is not an official OS scheduler (why?), and the OS is still making scheduling decisions on top of your algorithm. Our implementation can be seen as a user-level (why?) process coordination mechanism that approximates scheduling algorithms.
  - We have to make sure that this approximation is as close as possible to the actual OS scheduling. What can you do to achieve this? HINT: make sure that the OS does

not need to manage too many additional processes in the background.
  - Please also see the man pages for `kill()`, `usleep()`, and `waitpid()`.

In addition to the provided Round Robin implementation, **you have to implement the following scheduling algorithms**:

- Shortest job first (SJF)
- First come first serve (FCFS – you can assume that the processes arrive in the beginning in the same order they are created)
- Multi-level feedback queue (MLFQ) with two levels of queues, where the first level queue uses Round Robin algorithm with a fixed time quantum, and the second level queue uses FCFS. If a process finishes its allocated quantum in the first queue, it is moved to the second queue. Processes are not allowed to move to the first queue from the second queue.

and compare their performances using **average response time (of 4 child processes)** as the evaluation metric.

Note that additional parameters, if required, for these algorithms will be provided in the report section below.

Important Notes:

1. You should analyze the given sample C file very well to understand how each process can be paused and resumed, as well as how timing of each scheduling decision can be ensured (e.g., how we make sure that a process uses its allocated processor time).
2. You need to devise a method to calculate the response time for each child process. Note that response time is different from execution time.
3. You should find out and list the assumptions you make to implement the additional scheduling algorithms. For example, what assumptions do you need to make about the processes and/or the system to implement your FCFS algorithm?
4. You may need to implement additional data structures to help with your scheduling algorithm implementations (e.g., for MLFQ).

Report

Your report should include the following:

Detailed explanation of implementation of different algorithms:

- How you changed the sample algorithm to implement the others,
- What assumptions you had to make for your implementations,
- What data structures, if any, you had to implement,
- How you are calculating/reporting the response time of each process using each scheduling algorithm,
- How you are devising each of the experiments described below.

Results of the following experiments:

- **Experiment 1: Best time quantum value for Round Robin:** You are initially given a time quantum of 1000 microseconds for the sample Round Robin implementation, where the workload sizes are as follows:
  - `WORKLOAD1 = 100000`
  - `WORKLOAD2 = 50000`
  - `WORKLOAD3 = 25000`
  - `WORKLOAD4 = 10000`

  You need to find the optimal time quantum value in your system given these workload sizes. Note that the optimal value should give the smallest average response time among the child processes. Your experimental design should include what quantum values you analyzed, why you chose those values, and the quantitative results to support your claims. Also, answer the following questions:
  - Why do you observe different average response time values for different time quantum values?
  - Would it help to give a specific quantum value for each process? Why or why not? You do not have to report numerical results for this. Text explanation is sufficient.
- **Experiment 2: Best time quantum value for MLFQ:** In this experiment, you are to find the best time quantum value for the first-level queue (that implements Round Robin), using the initial process workload sizes, where the workload sizes are as follows:
  - `WORKLOAD1 = 100000`
  - `WORKLOAD2 = 50000`
  - `WORKLOAD3 = 25000`
  - `WORKLOAD4 = 10000`

  Note that the optimal value should give the smallest average response time among the child processes. Your experimental design should include what quantum values you analyzed, why you chose those values, and the quantitative results to support your claims.
  - You have to also see if your selected time quantum value leads to a Round-Robin-only or an FCFS-only algorithm (i.e., if the quantum value is too big or too small). If that is the case, you have to explain whether such a setup is good or bad.
  - Also, answer the following question: Would it help to give a specific quantum value for each process in the first-level queue? Why or why not? You do not have to report numerical results for this. Text explanation is sufficient.
- **Experiment 3: Comparing the performance of all algorithms:** In this experiment, you will have two cases:
  - **Case 3a:** You will compare the performance of Round Robin, SJF, FCFS, and MLFQ using the default workload sizes. For Round Robin and MLFQ, you should use the quantum values from Experiments 1 and 2. The performance comparison should be based on average response time of all child processes. You should explicitly answer:
    - Which scheduling algorithm is the best in terms of average response time? Does this outcome support the theoretical information we saw in class?
  - **Case 3b:** You will repeat Case 3a, with the following workload sizes:
    - `WORKLOAD1 = 100000`
    - `WORKLOAD2 = 100000`

        - `WORKLOAD3 = 100000`
        - `WORKLOAD4 = 100000`

    You should explicitly answer:

        - Which scheduling algorithm is the best in terms of average response time? How does that compare to Case 3a?
        - Would it make sense to use different time quantum values, as compared to Case 3a, for Round Robin and MLFQ? Why or why not?

- NOTE: Your experimental cases should not include just a single experiment for each case. You need to make sure that you are repeating each case enough times to have statistically significant results.
    - How many times did you repeat each case?
    - Report average and standard deviation and compare how standard deviation changes among these different cases. Explain why you observe higher standard deviation in some cases, if applicable.

25% Additional Credit: Quantify and report the context switch overhead in each experimental case (1, 2, 3a, and 3b). Note that to be eligible for this additional credit, you should both

- Explain your methodology to estimate the context switch overhead and
- Provide numerical context switch overhead results.

Deliverables

Your assignment submission must include TWO files:

- A **zip** file containing one or multiple C files (depending on your implementation) representing your scheduling algorithm implementations along with a README file that shows what each C file corresponds to.
- Project report containing your implementation details and discussing your results. Please submit your report in DOCX or PDF (preferred) format.

Your final score will be determined by the combination of your report and turning in functional C file(s) on time. **Note that the report should be at most 4 pages long. If you are including extra credit results, your report can be at most 5 pages.**