COMP2432 Group Project (2018/2019 Semester 2)

Submission deadline: April 4, 2019

Weighting: 15%

Project title: **S**tudy **S**cheduling **S**ystem (**`S3`**)

Scenario

During the semester's end (the last two weeks before exam), there are usually many individual assignments and group projects which have to be completed on or before the examination. In addition, students want to have sufficient time to do revisions of the lectures. Alice is a year two student and has subscribed to six subjects in this semester. She wants to manage her time to study and to do the assignments in an effective way. According to her experience from the previous year, she can squeeze out a few hours a day to do those assignments and projects and still have the time for study revisions. Of course, there are still classes that Alice has to attend and some extracurricular activities that she does not want to miss. Knowing that you are studying the subject, COMP2432, and have learnt how to schedule tasks to get a best overall performance, you are invited to help. Your mission is to help Alice to prepare a schedule/timetable to arrange those tasks so that she can achieve the best result in spending the valuable time.

Project Requirements

In this project, it is an opportunity to apply the theory of process scheduling which you have learnt from COMP2432 to a daily-life scenario and to produce a **Timetable**. The project simply consists of four parts:

Part 1. Write a program that allows user to add details of assignments and/or classes' details and/or personal activities (duration, subject, priority, etc) to the scheduler. For each input item, you need to assign an ID and record down whether the item is "accepted" or "rejected" in a log file based on different algorithms used (`S3_xxx.log, where "xxx" should be the algorithm used`). This is referred to as the **Input Module**.

Part 2. Extend your program developed in *Part 1* with a scheduler to generate timetables. The scheduler may implement different scheduling algorithms which are similar to those covered in lectures. This is the **Scheduling Kernel**, within the **Scheduling Module**.

Part 3. Augment the program with the facilities to print out timetables for Alice nicely according to the chosen algorithm in *Part 2*. This constitutes the **Output Module**.

Part 4. Provide the program with the ability to generate a summary report and analyze the results produced in *Part 3*. Compare the different timetables (generated from different algorithms) and find out which one would be the most suitable one for Alice. By the way, an outstanding list may be included in this report for those tasks cannot be scheduled. This final module is the **Analyzer Module**.

You must form a group of 4 to 5 (at most) persons for the project. The project must be implemented using **programming language C** and it must be successfully executed on **ANY ONE of the Linux Servers** in the Department of Computing. You will have to specify to us on which Linux server your project would be executed (e.g. **apollo**, **apollo2**).

***** *Note that "`gcc`" is the only compiler that we are going to use in this project.* *****

Implementation

User Interface

A user interface is needed. For user to input commands and to add/create those assignments and activities in the scheduler (`S3`). The input methods may look like the following (*blue colored lines*).

```
   ~~WELCOME TO S3~~

Please enter:
> addPeiod 2019-04-08 2019-04-21 19:00 23:00
Please enter:
> addAssignment COMP2432A1 2019-04-18 12
Please enter:
> addProject COMP2422P1 2019-04-20 26
Please enter:
> addRevision COMP2000 2019-04-14 19:00 2
Please enter:
> addActivity Meeting 2019-04-18 20:00 2
Please enter:
> addBatch S3_tasks_00.dat

…
…

Please enter:
> runS3 FCFS S3_report_fcfs_01.dat
Please enter:
> runS3 PR S3_report_pr_02.dat
…
…
Please enter:
> runS3 RR S3_report_rr_09.dat
…
…
Please enter:
> exitS3
Bye-bye!
```

Input Format

For the inputs, most likely they all could be treated as integers and we simply assume there are no errors for those input values.

No matter whether inputs are provided line by line through the program or imported through an external file in plain text format, you are recommended to input a large number of events which exceed the capacity of the available time slots in your testing so as to produce some outstanding list. You are also recommended to test for fewer tasks to check whether or not the algorithms implemented work as expected.

Input	addPeiod 2019-04-08 2019-04-21 19:00 23:00
Format	**addPeriod** [start date] [end date] [start time] [end time]
Usage	It is to specify the period (dates of start and end) and the time slots [start time and end time] for scheduling events. Date and time format is year-month-day and hour (YYYY-MM-DD and hh:mm, simply take the hour and ignore the minute (i.e. mm=00) and use the 24-hour). Actually, this should be the first command for the application.

Input	addAssignment COMP2432A1 2019-04-18 12
Format	**addAssignment** [subject code with assignment number] [due date] [duration]
Usage	[addAssignment] is to add a subject assignment to the scheduler. Follows by the due date and hours required to finish the assignment. The format [YYYY-MM-DD] is that YYYY is year, MM is month and DD is day. Then the [duration] is the number of hours that required to finish the assignment. Since there are only 4 timeslots per day that a task could be allocated, for a task whose duration is more than 4 hours it could be allocated to the timeslots of the next coming days. However, for assignment, you may not finish it at one time. For example, there is an assignment which needs 4 hours to complete and should be done before Day_6. You may separate the assignment into 4 sections (require 4 individual timeslot). In the other word, you divide that one request into four different requests. Do the first section in Day_1, Timeslot_1, the second section in Day_1, Timeslot_4, the third section in Day_3, Timeslot_2 and finally complete the fourth section in Day_5, Timeslot_3. Moreover, that may cause some assignments which could not be fully completed, i.e. less than 100 percent. If so, you should indicate the percentage of completion in the report.

Input	addProject COMP2422P1 2019-04-20 26
Format	**addProject** [subject code with project number] [due date] [duration]
Usage	[addProject] is similar to [addAssignment]. It is to add a subject project to the scheduler. Follows by the due date in the format [YYYY-MM-DD]. Then the duration is specified (hours).

	If you are considering to use **"priority"** as one of the algorithms that to be implemented in your application, [addProject] would have a higher priority than [addAssignment].

Input	addRevision COMP2000 2019-04-14 19:00 2
Format	**addRevision** [subject code] [date and time] [duration]
Usage	[addRevision] is to add a revision section to the scheduler. Revision date in the format [YYYY-MM-DD hh:mm]. Then the duration is specified (number of hours required). Unlike the previous two, this task should be done in one off. If the task cannot be done in one go, simply reject it. For considering **"priority"**, the order looks like the following. [addProject] → [addAssignment] → [addRevision] → [addActivity].

Input	addActivity Meeting 2019-04-18 20:00 2
Format	**addActivity** [name of the event] [date and time of the event] [duration]
Usage	[addActivity] is similar to [addRevision]. Instead of the "subject code", here we use "name of the event". Then also it follows by the date and time of the event in the format [YYYY-MM-DD hh:mm]. And then, the duration.

Input	addBatch S3_tasks_01.dat
Format	**addBatch** [filename]
Usage	[addBatch] allows user to prepare a text file which contains multiple lines of different requests and import into the program.

Input	runS3 FCFS S3_report_fcfs_01.dat
Format	**runS3** [algorithm] [filename]
Usage	[runS3] is to generate a schedule with the algorithm specified (in the example "FCFS" means " first come first served"). There are other algorithms that can be used: - PR, priority - SJF, shortest job first - SRT, shortest remaining time - RR X, round robin with X is the number of time slots (quantum) to be assigned in each round. - etc For the output file, for example S3_report_fcfs_01.dat, it is just a text file that include the time table and the analysis report (Output Module and Analyzer Module). In addition, the acceptance or rejection of a request should be recorded in a corresponding log file (stored in Input Module).

Input	exitS3
Format	**exitS3**
Usage	[exitS3] is to terminate the program properly.

To ease your work, we make the following assumptions and notes:

1. The "period" of the scheduler is (tentatively) from April 8 to 21 (14 days). And for each day, it starts from 19:00 and ends at 23:00 (4 hours). Indeed, your program should be able to handle more than 14 days and 4 hour timeslots.
2. One time slot is assumed as one hour, and inputs are simply to indicate hours.
3. As "project" usually carries more marks than "assignment", we should finish "project" first. Furthermore, if we could submit a complete project or assignment, it should be the best. In some cases, we might submit partial completed assignment and/or project to get partial marks. If it happens in your application, you should indicate the completion percentage of that assignment or project.
4. To allow more time slots to complete assignments and projects, we have the priority setting (if applicable) as following: Project → Assignment → Revision → Activity.
5. There are many implementation methods for the modules. The scheduler module may be implemented as a separate process, in the form of a child process created by the parent via **fork()** system call. The output module can also be a separate process. The scheduler may also be implemented as a separate program. If the parent is passing activities' details to a child scheduler, one should use the **pipe()** and associated **write()** / **read()** system calls. If the parent is passing information to a separate scheduler program, one could use the Unix shell **"pipe"** (denoted by **"|"**).
6. Both **pipe()** and **fork()** system calls are very important in operating systems. If both are used properly in the program, the maximum possible grade is (A). On the other hand, if both system calls are not used in the program, only a passing grade (D or D+) would be awarded. Intermediate scoring will be given if only one system call is used, depending on the extent of usage.
7. Bonus would be given if you can propose a better algorithm which is not the one of the existing algorithms taught in the class. You should provide evidences to proof your algorithm has a better performance.

Output Format

The timetable format may look like the one below (it is just for your reference, no need to follow it exactly). You may have your own version to print out the timetable.

Alice Timetable
Period: 2019-04-08 to 2019-04-21
Algorithm: FCFS

Date	19:00	20:00	21:00	22:00
2019-04-08	COMP2432A1	COMP2432A1	COMP2432A1	COMP2432A1
…	…	…0	…	…
…	…	…	…	…
2019-04-14	COMP2000	COMP2000	N/A	COMP2422P1
2019-04-15	COMP2422P1	…	…	…
…	…	…	…	…
…	…	…	…	…
2019-04-18	N/A	MEETING	MEETING	N/A
…	…	…	…	…
2019-04-21	…	…	…	…

The table simply indicates the "date", "time slots" and the activities assigned. Here "N/A" means the timeslot has NO assignment.

Also, the summary report looks like the following one.

```
***Summary Report***

Algorithm used: XXXXXXXXXXXXXXXXX

There are 9999 requests.

Number of request accepted: 9999
Number of request rejected: 9999

Number of time slots used:  9999 (999%)
```

In this part, it is simply to show the summary of the performance of the algorithm being used. That is, number of events that have been accepted or rejected by the system should be displayed. In addition, number of time slots used and the percentage of usage are shown.

```
***Log File - FCFS***

ID     Event                                          Accepted/Rejected
=====================================================================
0001   addAssignment COMP2432A1 2019-04-18 12          Accepted
0002   addProject COMP2422P1 2019-04-20 26             Accepted
0003   addRevision COMP2000 2019-04-14 19:00 2         Accepted
0004   addActivity Meeting 2019-04-18 20:00 2          Accepted
0005
0006
…
…                                                      Rejected
…
```

For each algorithm, there is a corresponding log file to show the “Accepted” or “Rejected” of each request.

Error handling

Although the program does not need to check for the correctness of the syntax of the input, some other errors handling are required in the application. For example, if the input date and time is out of the "*period range*" entered, you should record it in an error log.

Documentation

Apart from the above program implementation, you are also required to write a project report that consists of following parts:

1. Introduction (*Why do you want to implement this project, i.e. the objective*?)
2. Scope (*What operating systems topics have been covered in this project*?)
3. Concept (*What are the algorithms behind*?)
4. Your own scheduling algorithm (*if any*)
5. Software structure of your system
6. Testing cases (*What have you done to test the correctness of the program/application*?)
7. Performance analysis (*Discuss and analyze the algorithm used in the program. Why it is better than the others*?)
8. Program set up and execution (*How to compile and execute your project? On which Linux server would you like your project to be executed*?)
9. *Appendix I* – source code, and, soft copies of "Time Table", "Summary Report" and "Log File".
10. *Appendix II* – **Contribution of Work**. You should indicate/describe each project group member's effort/workload for the project in details. Individual assessment would be based on this part mainly.

Demonstration

The demonstration will be held in **Week 11** (*Saturday and Sunday*, tentatively). Each group has 15 minutes to demonstrate the implementation of the scheduler. Please write down the *name*s and *student ID*s of your group members on the **COMP2432 Project Group Form** which should be posted outside **PQ732**.

Please confirm your group on or before **February 28, 2019**.

Otherwise, you will be assigned to a group randomly. Demonstration schedule/timetable would be released once all the groups' details are collected. The demo time slot would be allocated to each group randomly.

Late Submission Penalty – **10 %** per day.

Submission

You must submit the following files to the Blackboard System:

1. Readme file – write down your project title and the name of each member in your group, together with all necessary information that may be required to run your program; name the file as "**Readme.txt**".

2. Source code and output files

   - Name of the source code file should be "**S3.c**"
   - Name the output file, as mentioned, "**S3_report_xxxx.dat**" where "**xxx**" is the name of the algorithm used.
   - Name the log file, "**S3_xxxx.log**" where "**xxx**" is the name of the algorithm used.

3. Name the project report as "**Project_Report_GXX.doc**" where "**GXX**" is the group number assigned.

4. Name the imported data file as "**s3_data.dat**" (*if applicable*).