COMP2432 Group Project (2021/2022 Semester 2)

Submission deadline: April 8, 2022

Weighting: 15%

Project title: **Project Meeting Scheduler** (**PMS**)

Scenario

PolyStar Interior Design Company plays the role as a commercial office interior design and contracting specialist in the market. There are designers/staff working in different project teams to supervise/coordinate a number of interior design and fit-out projects. Due to the developmental need of different projects concurrently, it is impractical that each member is only working on a single project for the entire period. In other words, each designer/staff member should be working on multiple projects simultaneously.

There are two roles associated with each project: *manager* and *member*. Each project has one and only one manager and a few members. The manager is in charge of the project, playing the role of a leader. Each designer/staff could be the project manager or project member of a few different projects. To ensure the quality of the projects, PolyStar does not want to overload the designers or staff members and provides a guideline on their maximum workload.

> *Designer or staff member could participate in **at most three projects** but could be the manager for **only one project**. For example, Staff_A could be the project manager of Project_X. He/She could not be the manager of another project Project_Y but could be a member of the other two projects. In addition, each project group could have **four staff members at most (i.e. one project manager + 3 project members)**.*

Project managers usually have meetings with their team members and/or clients. However, most of them are busy enough and could not arrange/manage those meetings properly and in time to fix problems. That had caused lots of problems in the progression of the projects. Some of them could not be solved in time and resulted in a loss to the company.

Knowing that you have learnt different scheduling algorithms from the operating systems course, the manager in PolyStar would like to offer you a freelance job. Would you mind helping PolyStar to create a "Project Meeting Scheduler" (*PMS*) so that there is a better utilization of human resource, with an improved meeting scheduling/rescheduling method? The goal is to implement a better meeting scheduler to reduce the problems in the progress of projects to ensure the scheduling and conduct of timely meetings.

Project Requirements

In this project, you will be given an opportunity to apply the theory of process scheduling you have learnt from COMP2432 to a daily-life scenario and develop the **Project Meeting Scheduler** (**PMS**) for PolyStar to deploy. A modular design should be adopted in this freelance job to enhance flexibility and maintainability. In general, this PMS system should be designed with modules with different functionalities. Furthermore, this project should be carried out in smaller steps, consisting of four parts, perhaps up to six parts:

Part 1. Develop a program PMS that allows users to create project teams (*one project manager + up to three members*) and add details of a meeting (*date, time, duration, and/or callees*) to the schedule for existing project teams. Besides the standard line by line input approach, PMS should be able to read in a batch file containing the information of meeting requests, i.e. one or more than one meeting requests are stored in such a batch file. Note that the one who initiates for the booking is called the "**user**" or the "**caller**" (*usually it should be the project manager*) and those team members involved in the meeting are called "**callees**". This program is referred to as the **Input Module**. Remember that the program should allow batch input, i.e. read in files for entries. It is recommended to store all meeting requests in a log file for future audit purpose. By the way, the application is developed to be used internally for the time being. If the result is satisfied, PolyStar will consider promoting to their clients to use the application to have better communications with their customers.

Part 2. Augment the program PMS to check whether or not the attendees (*team members*) are available. If so, it assigns a reservation on each member's timeslot(s). If not, the request would be rejected. For example, if a staff person has scheduled a timeslot for a meeting, he/she will reject the other meeting request which conflicts with the scheduled timeslot. However, if the priority of projects is considered in your algorithm, the higher priority one might overwrite the lower priority one and make a revision to the lower priority meeting schedule in the form of rescheduling, or a rejection to the lower priority meeting schedule in the form of cancellation. This latter arrangement is like rejecting the existing one but accepting the new one. Of course, you may have other assumptions on how the conflicting meetings are being scheduled or handled. For example, you may consider postponing the previous/new request, whichever is of a lower priority, to a new timeslot when all members are available. You might consider to include some more "humane" criteria in scheduling, e.g. ensuring that there would be some lunch time before or after a lunch-time meeting, and that the total number of meeting hours per day does not exceed a certain limit, e.g. 5 hours.

Part 3. Extend the program developed in *Part 2* with a scheduler to generate a meeting schedule for all meetings of each staff member or project team (*e.g. Staff_A's Meeting Schedule or Team_A Meeting Schedule*). The scheduler may implement several scheduling algorithms like those covered in lectures. (*In this project, you are asked to implement **at least two**. One of them should be the relatively straightforward First Come First Served*.) This is called the **Scheduling Kernel**, within the **Scheduling Module**.

Part 4. Augment the program with the facilities to print/export meeting schedule for each of the staff members or Project Teams in *Part 2*. Not only the accepted requests are included but also those rejected or rescheduled requests should be included in the report too. This constitutes the **Output Module**.

Part 5. (*Bonus up to 2%*) Analyze the program with different scheduling algorithms which you have tested for. List out those advantages and disadvantages which can be achieved from your program implemented.

Part 6. (*Bonus up to 8%*) Implement the attendance of each team member and discuss on the use of this report. For example, is it possible to use this report to appraise the performance of the staff member and how?

You may form a group of 3 - 5 persons for this project. The project must be implemented using programming **language C** and it must be successfully executed on **ANY ONE of Linux Servers** (*apollo or apollo2*) in the Department of Computing.

We accept only the C-source code file(s). We simply compile your source code file(s) and then execute the compiled program. If you use any additional library files in your application, you should have ***sought the approval*** from us first. Those additional library files should be included in your submission and declare clearly how they are applied in your application.

***** *Note that we use only the "gcc" or "cc" compiler to compile your program. In other words, if your program cannot be compiled by the compiler of department's servers, apollo or apollo2, we simply treat yours as a* ***failed program***.

Implementation

User Interface

First, your program should provide a menu to allow users to choose which function would be used, for example, to create Project Teams or to input meeting request or to print reports, etc. Below is an example for reference.

```
   ./PMS

   ~~ WELCOME TO PolyStar ~~

1.  Create Project Team

2.  Project Meeting Request
    2a.  Single input
    2b.  Batch input
    2c.  Meeting Attendance            (if implemented)

3.  Print Meeting Schedule
    3a.  FCFS (First Come First Served)
    3b.  XXXX (Another algorithm implemented)
    3c.  YYYY (Attendance Report)      (if implemented)

4.  Exit

  Enter an option: __
```

*** It is quite a common arrangement or norm to enter "0" to return to the main menu (*no matter where you are, even though you are at the main menu already*).

Command Syntax and Usage

To execute the program	
Syntax	**./PMS**
Use	Simply to enter [**./PMS**] to start the program.

Menu Part 1	**1. Create Project Team**
Syntax	**Enter> Team_A Project_A Alan Cathy Fanny Helen** **>>>>>> Project Team A is created.** **Enter> 0**
Use	The first [Enter] line is to create a project team, Team_A for the project (*Project_A*) and Alan is the project manager. Members are Cathy, Fanny and Helen. The second [Enter] line that has the input "**0**" is to ask the application to return to the Main Menu.

Menu Part 2	**2. Project Meeting booking** **2a. Single input** **2b. Batch input** **2c. Meeting's attendance**
Syntax	**For 2a, Team_A 2022-04-25 09:00 2** **For 2b, batch01_Requests.dat** **For 2c, xxx**
Use	This part is about the creation of meeting schedule. It is to collect the meeting requests and schedule the meetings for project teams. For 2a, a meeting request is being added for Team_A on April 25, 2022 at 9 o'clock and the duration is 2 hours. Application may simply reply that the request is recorded (*not yet scheduled, i.e. accepted or rejected*). Those requests would be scheduled when one schedule algorithm is executed to generate the schedule. For 2b, similar requests are stored in one text file; here as an example, it is named "batch01_Requests.dat". Also, the application may make a simple reply to acknowledge the input. However, it is recommended to copy all requests in the file to the application. (*Again, you may have your way to process the batch input file(s).*)

	For 2c, since it is an optional function to be added in your application, you may have your own design to do it. However, you need to explain your assumptions, for example, what kind of data to be input and what kind of the analysis report would be generated for what purpose(s), etc. Enter "0" (zero) to return to the main menu.

Menu Part 3	**3. Print Meeting Schedule** **3a. FCFS (First Come First Served)** **3b. XXXX (Another algorithm implemented)** **3c. YYYY (Attendance Report)**
Syntax	**For 3a, FCFS 2022-04-25 2022-04-27** **For 3b, XXXX 2022-04-25 2022-04-30** **For 3c, YYYY 2022-04-28 2022-05-02**
Use	In this part, we are going to print out/export the meeting schedule. For 3a, it is to print out the meeting schedule (*based on the FCFS algorithm*) to a file for the three specified days (*start from 2022-04-25 (yyyy-mm-dd) to 2022-04-27*). The application would prompt a message to indicate the name of the exported file, for example, **Printed. Export file name: Schedule_FCFS_01.txt** For 3b, similar to 3a but the "XXXX" is to be replaced by the abbreviation of the name of the algorithm you have implemented. Also, "Start Date" and "End Date" are needed to define the period. For 3c, the "YYYY" is replaced by the abbreviation of the name of the attendance report that you have implemented. Again, "Start Date" and "End Date" are required.

Menu Part 4	**4. Exit**
Syntax	**Enter an option: 4**
Use	Simply enter the option **4** to exit and terminate the program.

To ease your work, we have the following **assumptions**:

1. There are staff members: Alan, Billy, Cathy, David, Eva, Fanny, Gary and Helen.
2. Five project teams are to be created for this project. That means there are five projects in progress.
3. Period of the meeting schedule: 2022-04-25 to 2022-05-14.
4. Working Day: Monday to Saturday
5. Timeslot: one hour per slot and starts from 09:00 to 18:00. It is applied to all working days.
6. Meeting would be started as early as 09:00 and ended at 18:00. That means request can be made to reserve timeslots which are available on that day only.
7. According to Assumptions 3 to 6, each working day should have 9 timeslots available and there are a total of 162 timeslots to be filled in for the testing period. You are recommended to overflow the time slots in your own tests on the program and see whether your program can handle different situations.
8. You may consider that those staff members and project teams would be represented by [**`child`**] processes. The parent process represents the PolyStar to host all the meeting requests and can invoke different algorithms to schedule/reschedule meetings, via communication with different child processes.
9. If a "priority" algorithm is applied, you may simply consider the alphabet order of the name of the project teams (or project manager, etc). For example, a "Team_B" may "displace" an already scheduled "Team_X" so that the caller/callee(s) involved would attend the meeting of "Team_B" and the meeting of "Team_X" should be canceled or reschedule (*if you have implemented the reschedule algorithm*). That could turn a previous "**Accepted**" status (displayed when the request is successfully scheduled early on) to a "**Rejected**" one (overtaken by this more important "Team_B" event).
10. 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 implemented as a separate process. The scheduler may also be implemented as a separate program. If the parent is passing request details to a child scheduler *process*, 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 "`|`").
11. Since process management and inter-process communication are important concepts in an operating system, you should attempt to make use of the **`pipe()`** and **`fork()`** system calls in your program. If **`pipe()`** and **`fork()`** system calls are both used properly in the program and also the re-scheduling method/algorithm, the maximum possible mark is 100% plus up to 10% of bonus points. On the other hand, if both system calls are not used in the program, only a passing mark would be awarded at best. Intermediate scoring will be given if only one system call is used, depending on the extent of usage.

Output Format

The “meeting schedule” (*example: Schedule_FCFS_01.txt*) may look like the following.

```
*** Project Meeting ***

Algorithm used: FCFS
Period: 2022-04-25 to 2022-04-27

Date            Start   End       Team          Project
================================================================================
2022-04-25      09:00   11:00     Team_A        Project_A
2022-04-25      12:00   16:00     Team_C        Project_C
…
…
…
2022-04-27      09:00   12: 00    Team_E        Project_E
…
…
…


================================================================================
Staff: Alan

Date            Start   End       Team          Project
================================================================================
2022-04-25      09:00   11:00     Team_A        Project_A
…
…

================================================================================
Staff: Cathy

Date            Start   End       Team          Project
================================================================================
2022-04-25      09:00   11:00     Team_A        Project_A
…
…

================================================================================

                                    - End -
```

```
*** Meeting Request – REJECTED ***

There are 999 requests rejected for the required period.

================================================================================
1.   Team_D 2022-04-25 09:00 2
2.   …
3.   …
…
…
999.  …
================================================================================
```

For the “Summary” may have the format as follows.

```
Performance:

Total Number of Requests Received: 999 (99.9%)
      Number of Requests Accepted: 999 (99.9%)
      Number of Requests Rejected: 999 (99.9%)

Utilization of Time Slot:

     Accepted request          - 99.9%

     Team_A                    - 99.9%
     Team_B                    - 99.9%
     …
     …

     Staff_A                   - 99.9%
     Staff_B                   - 99.9%…
     …
     …

Attendance:

     Staff_A                   - 99.9%
     …
     …
```

Error handling

Although the program need not check for the correctness of the format and values that users input, some other error handling features are required in the application. For example, there are requests with the duration hours over the time allowed. That is, in one day a request may only set the duration from 1 to 9. If there is a request on a meeting occurring in duration 10, your application should be able to treat this as an invalid request. That erroneous request would not be counted in neither “Accepted” nor “Rejected”.

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 have this project* (*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 (*In the report, briefly describe what you have done to test for the correctness of the program? For the details of the tests, that part should be attached in the section "**Appendix**"*.)
7. Performance analysis (*Discuss and analyze the algorithms used in the program, if there are more than one algorithm. Why is the one you proposed 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. Individual member contribution – fill out the following table. For Item 1, fill in a task list assigned and done by the member. For Item 2 to 7, each box, fill in one of following letter grades:
   A – excellent
   B – good
   C – satisfactory
   D – marginal satisfactory
   F – fail

   According to your inputs, we will consider adjusting your individual grade. That means each member may receive a different grade eventually. If a member who has no contribution (*i.e. has done nothing*), that member might receive a ZERO finally.

Item	Group: **GXX**	Member 1: A	Member 2: B	Member 3: C	Member 4: D	Member 5: E
1	Responsibility: List the task(s) that each individual was (*were*) responsible for.	1. xxx 2. xxx 3. xxx	1. xxx 2. xxx 3. xxx	1. xxx 2. xxx 3. xxx	1. xxx 2. xxx 3. xxx	1. xxx 2. xxx 3. xxx
2	Cooperation: Able to work within team; willingly performed task(s).					
3	Punctuality: On time for team meetings.					
4	Reliability/Dependability: Performed tasks within established times.					
5	Evaluative: Offer constructive criticism and helpful evaluation of work.					
6	Creativity: Provide meaningful insight to project team.					

Item	Group: **GXX**	Member 1: A	Member 2: B	Member 3: C	Member 4: D	Member 5: E
7	Overall Effort: Overall contribution to project.					

10. Appendix – source code, testing data, testing results, and copies of the samples of "project meeting schedule" and "rejected list".

****Noted that you should use a proper report format.*

Demonstration

The date of demonstration is tentatively scheduled on April 9, 2022. The time is from 12:00 pm to 6:00 pm (*six hours*), please reserve your time on that day. There are two parts in the demonstration. The first part is to make a video (*in MP4 format*) to introduce and demonstrate your application with a dataset which is prepared by your group (*length of the demonstration video is about 3 to 5 minutes*). Submit the first video together with the source code and the project report on or before the due date (*April 8, 2022*).

The second part is to have another video (3 to 5 minutes) to show the execution of your application based on the dataset that is provided by us. (*The Testing Dataset would be released just before the demonstration*.) A set of documents of the running results (*which are generated by your application*) should be submitted to the Blackboard once again (*details would be provided later through the Blackboard*). If your application cannot run the provided data normally or successfully, you are allowed to **correct/modify** your application within a grace period. The revised/modified source code should be submitted to the Blackboard again together with the second video before the end of the demonstration (*i.e. 6:00 pm*). In addition, you should let us know what error(s) you have encountered and how you fix the problem.

****Note: All members in the group should show up in the two videos, no matter whether or not they have been assigned task(s) to do. Also, you have to let us know who you are properly in the video for individualized assessment.*

In addition, if the size of the video is too large and causes the video cannot be uploaded to the Blackboard, you may send us a link to download it.

Mark distribution

The mark distribution of this project is as follows:

Implementation:	60 %
Documentation:	25 %
Demonstration:	15 %
Bonus:	10 %

Bonus

The bonus marks will be awarded for being able to implement the function of tracking "attendance" of project members and use those records to give useful advice to the company. Of course, you need to suggest/assume the necessary pieces of information (*data*) to be input into or used in the system. (*An extra demonstration would be required if there is a need.*)

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

Submission

You must submit the following files (*zip them all in one file, except Item 1 – Weekly Progress Report*) to the Blackboard System:

1. Weekly Progress Report: Each student should submit one Weekly Progress Report to indicate the progress of the project of their part. It is simply to write one or two sentences to declare the percentage of completion and/or the problem(s) that you encountered, etc. Note that you should report about yourself, not about other members.

2. 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* ***"GXX_Readme.txt"***).

3. Source code and output files

   - name the source code file as **"GXX_PMS.c"**
   - name the output files:
     - Project Meeting Schedule, for example, you have implemented three different algorithms, the outputs may look like the following (*just a reference*),
       "GXX_FCFS_Schd_01.dat" for the FCFS algorithm,
       "GXX_PRIO_Schd_02.dat" for the Priority algorithm, and
       "GXX_OPTI_Schd_03.dat" for the Optimized algorithm, etc. The number used in the file name is to keep track of how many output files have been printed, i.e., for example, FCFS meeting schedule might have been printed more than one time at different time periods.
     - Summary (*Performance*), like Project Meeting Schedule,

4. Name the project report as **"GXX_PMS_Report.docx"**.

5. Record all inputs (*testing data*) used in the testing cases and save it to **"GXX_tests.dat"**.

   * GXX – your group number assigned

   * Group these files (*except Item 1 – Weekly Progress Report*) in **ONE zipped file** and upload the zipped file to the Blackboard on or before the due date.

6. The first video file (*in MP4 format*): name it as ***"GXX_video.mp4"***. It is to introduce and demonstrate your application in 3 to 5 minutes. Also, submit this file on or before the due date.

7. The second video file together with the required documents: name the video file as ***"GXX_demo.mp4"***. The video is to illustrate how your group completed the demo tasks. In addition, there are a set of documents which are generated by your application to be submitted before the end of the demonstration.

8. Revised source code file(s) (*if applicable*) is to be submitted before the end of the demonstration.