COMP2432 Group Project (2020/2021 Semester 2)

Submission deadline: April 15, 2021

Weighting: 15%

Project title: **R**oom **B**ooking **M**anage**r** (**RBM**)

Scenario

PolySME Business Center is a company, which supports the concept of shared offices and provides office facilities to small business companies, especially those one-man companies. The company offers general office facilities such as telephone lines, fax lines, LAN/Wi-Fi network, meeting rooms, etc. PolySME finds that there is a problem with its current meeting room booking system designed 10 years ago. It is not flexible enough to make good utilization of the meeting rooms and to satisfy most requests from its tenants. In PolySME, there are three meeting rooms in use currently. The three rooms are in different sizes; two are small, allowing 10 persons (*at most*) and the third one is the largest, capable of holding 20 persons (*at most)* in a meeting. In addition, PolySME provides add-on/special facilities for each meeting room, for example, projector + screen for presentation/meeting use, or webcam + monitors for online conferences or meetings, to be set up in the rooms on demand. Furthermore, such additional facilities come with different configurations. For example, different projects/webcams support different resolutions, and the screens may be in different sizes. Currently, tenants of PolySME may simply make a booking of a meeting room or book a room equipped with special facilities or reserve only the meeting facilities. For example, Tenant_A makes a booking for a meeting room for 8 persons with a projector projector_2K with a screen screen_100. If there are both the availabilities of the room and the required facilities, the booking request is accepted. Otherwise, the booking request is simply rejected. For instance, all the projectors of type projector_2K might have been booked by other tenants so that the request is rejected. In the other words, that means the current system does not provide any alternative plan to satisfy or reschedule the booking when some of the requested item or location is not available for the requested time slot.

Knowing that you have learnt different scheduling methods from Operating Systems, the manager in PolySME is offering you a part-time job that you could use to satisfy your WIE. Would you mind to help PolySME to revise its room booking system to achieve a better utilization with an improved scheduling/rescheduling method? The goal is to improve the booking situation so as to increase company's revenue on renting these to its tenants.

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 produce the **Room Booking Manager** (**RBM**). The project consists of the following parts:

Part 1. Develop a program that allows users to add details of a booking (*date, time, duration, and/or callees etc.*) to the schedule. Besides the standard line by line input approach, RBM should be able to read in batch files which containing the

booking requests, *i.e. one or more than one booking requests are stored in such batch files*.

Note that the one who initiates for the booking is called the "*user*" or the "*caller*" and those others involved in the booking are called "*callees*". This part of the program module is referred to as the **Input Module**.

Part 2. Extend the program developed in *Part 1* with a scheduler to generate timetables for all bookings (e.g. meeting room booking schedule and tenant booking records). The scheduler will implement several scheduling algorithms similar to those covered in lectures. This is the called the **Scheduling Kernel**, within the **Scheduling Module**.

Part 3. Augment the program with the facilities to print booking schedule for meeting rooms and related devices in *Part 2*. Those rejected bookings should all be included. This constitutes the **Output Module**.

Part 4. Provide the program with the ability to generate a summary report to analyze the results produced in *Part 3*. Compare the different schedules (*generated from different algorithms*) and find out which one would make the best use of the three meeting rooms. Your program should preferably be able to process N meeting rooms (N = 3 in the existing case) to cope with the expansion of PolySME in the near future. It should also be preferably able to handle more resource types in future. By the way, an outstanding (*rejected*) list may also be included in this report for those requests that cannot be scheduled. This final module is the **Analyzer Module**.

Part 5. Augment the program RBM to check whether or not the required facilities are available. If so, it assigns a priority on each booking. For example, if a room is booked for a presentation and should be equipped with projector and screen, the projector and screen will be reserved for that booking with a higher priority. Even though the projector and screen have been reserved for another booking but without using a meeting room, i.e. someone reserves only the two pieces of equipment by that moment, the booking would be then rejected. Of course, you may have other assumptions on how the conflicting bookings are being scheduled/rescheduled, and the associated priority.

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

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

Implementation

User Interface

First of all, your program should provide an interface to allow the user to input the details of bookings and/or commands for the application. There are several input methods to be accepted by the system, and the system should be able to store all requests (*valid and invalid*). Below is an example of some inputs and outputs on a screen for reference.

```
./RBM
~~ WELCOME TO PolySME ~~
Please enter booking:
addMeeting –tenant_A 2021-05-10 09:00 2.0 8;
-> [Pending]
Please enter booking:
addPresentation –tenant_B 2021-05-14 08:00 3.0 12 projector_4K
screen_150;
-> [Pending]
Please enter booking:
bookDevice –tenant_B 2021-05-15 13:00 2.0 projector_2K;
-> [Pending]
…
Please enter booking:
addConference –tenant_E 2021-05-16 14:00 2.0 15 webcam_UHD monitor_75;
-> [Pending]
Please enter booking:
importBatch –batch001.dat;
-> [Pending]
…
Please enter booking:
bookDevice –tenant_C 2021-05-011 13:00 2.0 projector_4K;
-> [Pending]
…
Please enter booking:
printBookings –fcfs;
-> [Done!]
Please enter booking:
printBookings –prio;
-> [Done!]
…
importBatch –batch099.dat;
-> [Pending]
…
Please enter booking:
addMeeting –tenant_A 2021-05-16 10:00 3.0 9 projector_2K screen_100;
-> [Done!]
…
…
Please enter booking:
printBookings –ALL;
-> [Done!]
Please enter booking:
endProgram;
-> Bye!
```

Some other requests would be input into the system.

Schedule is printed which is based on the input bookings up to that moment.

Command Syntax and Usage

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

Command	**addMeeting**
Syntax	**addMeeting -aaa YYYY-MM-DD hh:mm n.n p bbb ccc;** **e.g. addMeeting –tenant_A 2021-05-16 10:00 3.0 9 projector_2K screen_100;**
Use	It is to add a booking for a meeting room with certain devices together. As in the example, [**tenant_A**] is to make a booking on [**May 16, 2021**] at [**10:00**], and the duration is [**3.0**] hours. In addition, it also requires a projector (**projector_2K**) and a screen (**screen_100**). [**-aaa**] – Tenant who makes the request. [**YYYY-MM-DD hh:mm**] – Date and time of the event, **YYYY**:Year (4 digits), **MM**:Month (2 digits), **DD**:Day (2 digits), **hh**:Hour (2 digits) and **mm**:Minute (2 digits). [**n.n**] – Duration of the appointment in hours (*one decimal place*) [**p**] – Number of persons to participate (*integer*) [**bbb**] and [**ccc**] – Additional devices to be required. (*Optional, but should be in pair when request, [projector]+[screen] or [webcam]+[monitor]*) **;** – End of current input. *Note*: RBM would assign a room which fits the requirements. For instance, RBM would assign either [**room_A**] or [**room_B**] since both are big enough to hold the 9 participants. Of course, RBM would also check the availability of the required devices (*projector and screen*). If all these are available, the request would be accepted. For [**addMeeting**], it is an optional choice to book any add-on facilities. Tenants may simply book a meeting room only.

Command	**addPresentation**
Syntax	**addPresentation -aaa YYYY-MM-DD hh:mm n.n p bbb ccc;** **e.g. addPresentation –tenant_B 2021-05-14 08:00 3.0 12 projector_4K screen_150;**
Use	Similar to [**addMeeting**], it is to make a booking for a presentation. As in the example, it is to add a request made by [**tenant_B**] who would conduct a presentation on **[May 14, 2021]** at **[08:00]** for **[3.0]** hours long. And, there would be 12 persons to take part in the presentation. In addition, the room should be equipped with [**project_4K**] and [**screen_150**]. [**bbb**] and [**ccc**] – Additional devices are required (*mandatorily include*). The use of parameters is same as the previous one.

Command	**addConference**
Syntax	**addConference -aaa YYYY-MM-DD hh:mm n.n bbb ccc ddd;** **e.g. addConference –tenant_E 2021-05-16 14:00 2.0 15 webcam_UHD monitor_75;**
Use	It is same as [**addPresentation**] but it would have a higher priority if you are to use the algorithm "priority". As in the example, [**tenant_E**] is to add a booking for a conference on **[May 16, 2021]** at **[14:00]** for **[2.0]** hours long. The room should be equipped with [**webcab_UHD**] and [**monitor_75**]. The use of parameters is same as the previous one.

Command	**bookDevice**
Syntax	**bookDevice -aaa YYYY-MM-DD hh:mm n.n bbb** **e.g. bookDevice –tenant_C 2021-05-011 13:00 4.0 projector_4K;**
Use	It is simply to reserve a specific device only. As in the example, it is to reserve for **[projector_4K]** on **[May 11, 2021]** at **[13:00]** for **[4.0]** hours long. That request is made by [**tenant_C**].

Command	**addBatch**
Syntax	**addBatch -xxxxx** **e.g. addBatch –batch001.dat**
Use	It is to read in a batch file, **batch001.dat** which is a plain text document and records one or more booking requests.

Command	**printBookings**
Syntax	**printBookings –xxx –[fcfs/prio/opti/ALL]** **e.g. printBookings –fcfs**
Use	It is to print the schedule (*Accepted and Rejected*) for users. In addition, a "Performance Report" is needed if [ALL] is used. The Performance Report is a summary of how many bookings are received and allocated, and how many are rejected in total that based on the algorithm used. Where **–[fcfs/prio/opti/ALL]** – Algorithms that would be used (*just a reference*). [**fcfs**] – First Come First Served [**prio**] – Priority [**opti**] – Optimized [**ALL**] – All algorithms being used in the application and the performance

	summary report. In the example, the "First Come First Served" is used. That is the first booking would be allocated to a time slot that other late requests which request the same time slot would just be rejected. For "Priority", where the one has a higher priority rank can take over the time slot (see **assumptions** below). That is, for example, a "conference" can replace an occupied time slot which has been assigned to a "Meeting". For "Optimized", actually there are different methods to produce an *optimized* result. Here, in a simple word, it is to reschedule those rejected appointments. Then, there would be a better performance to utilize the facilities in PolySME. In the other word, this part would be considered as the "*bonus*".

Command	**`endProgram`**
Syntax	**`endProgram;`**
Use	This simply ends the program completely, upon collecting all the **`child`** processes and closing all the files.

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

1. Input formats follow the examples and make no difference and we simply assume all the input formats are correct.
2. Components in RBM are limited to the following:
   - five tenants – **`tenant_A`**, **`tenant_B`**, **`tenant_C`**, **`tenant_D`** and **`tenant_E`**;
   - three meeting rooms – **`room_A, room_B`** and **`room_C`** (*room_A and room_B are of the small meeting rooms (<=10) while room_C (<=20) is the largest one*);
   - three webcams – **`webcam_FHD`** × 2 and **`webcam_UHD`** × 1;
   - three monitors – **`monitor_50`** × 2 and **`monitor_75`** × 1;
   - three projectors – **`projector_2K`** × 2 and **`projector_4K`** × 1;
   - three projection screens – **`screen_100`** × 2 and **`screen_150`** × 1.

   All these could be represented by [**`child`**] processes. The parent process represents the PolySME to host all the bookings and can use different algorithms to *schedule/reschedule* bookings.

3. If a "*priority*" algorithm is applied in the booking system, you may consider that a "*Conference*" has the highest priority because it contributes the most profit. Then, it is "*Presentation*". The third one is "*Meeting*". Lastly, it is "*Device*". In other words, a "*Presentation*" may "*displace*" an already scheduled "*Meeting*" so that the caller/callee(s) involved would attend the "*Presentation*" but the "*Meeting*" would be **canceled**. That could turn a previous "**Accepted**" status (*as the Meeting is successfully scheduled early on*) to a "**Rejected**" one (*overtaken by this more important "Presentation" event*).

4. 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 appointment 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 "**`|`**").
5. 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. Intermediate scoring will be given if only one type of system call is used, depending on the extent of usage.
6. We test the schedule for a period, from *10 to 16 May, 2021*. One time slot is an hour of time. That means each room should have 24 time slots a day and there are 7 days to be filled in. You are recommended to overflow the time slots in the tests.

Output Format

The “appointment schedule” – “ACCEPTED” and “REJECTED” may look like the following.

```
*** Room Booking – ACCEPTED / FCFS ***

Tenant_A has the following bookings:

Date          Start   End     Type           Room       Device
==========================================================================
2021-05-10    09:00   15:00   Meeting        room_A     *
2021-05-12    12:00   16:00   Presentation   room_B     projector_2K
                                                        screen_100
2021-05-15    09:00   12:00   Conference     room_C     webcam_UHD
                                                        monitor_75
…
…
Tenant_B has the following bookings:

Date          Start   End     Type           Room       Device
==========================================================================
2021-05-14    08:00   11:00   Presentation   room_B     projector_4K
                                                        screen_150
2021-05-15    12:00   16:00   Presentation   room_A     projector_2K
                                                        screen_100
2021-05-15    13:00   15:00   *              *          projector_2K
2021-05-16    10:00   15:00   Conference     room_C     webcam_UHD
                                                        monitor_75
…
…
   - End -

==========================================================================
```

```
*** Room Booking – REJECTED / FCFS ***

Tenant_A (there are 999 bookings rejected):

Date          Start   End     Type           Device
==========================================================================
2021-05-16    09:00   12:00   Meeting        -
2021-05-17    18:00   21:00   Presentation   projector_4K
                                             screen_150
…
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
…
Tenant_B (there are 999 bookings rejected):

Date          Start   End     Type           Device
==========================================================================
2021-05-13    13:00   15:00   Conference     webcab_UHD
                                             monitor_75
…
…

   - End -

==========================================================================
```

The “Summary” may have the format as shown below.

```
*** Room Booking Manager – Summary Report ***

Performance:

  For FCFS:
         Total Number of Bookings Received: 999 (99.9%)
               Number of Bookings Assigned: 999 (99.9%)
               Number of Bookings Rejected: 999 (99.9%)

         Utilization of Time Slot:
               room_A         – 99.9%
               room_B         – 99.9%
    …
               webcam_FHD     – 99.9%
    …
               projector_4K   – 99.9%
    …
    …

         Invalid request(s) made: 999

  For PRIO:
         Total Number of Bookings Received: 999 (99.9%)
               Number of Bookings Assigned: 999 (99.9%)
               Number of Bookings Rejected: 999 (99.9%)

         Utilization of Time Slot:
               room_A         – 99.9%
               room_B         – 99.9%
    …
               webcam_FHD     – 99.9%
    …
               projector_4K   – 99.9%
    …
    …

         Invalid request(s) made: 999
```

Error handling

Although the program does not need to check for the correctness of the format and values that are input by user, some other error handling features are required in the application. For example, if there are incorrect names of device, RBM should return an error message to indicate that and simply ignore that request.

Documentation

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

1. Introduction (*What is the objective of this project?*)
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. Program structure of your application
6. Testing cases (*In the report, briefly describe what you have done to test 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 one algorithm. Why the one you proposed 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 – source code; testing data, testing results, and samples of "room booking timetable" & "rejected list" which are generated by your application.

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

Demonstration

The date of demonstration is *tentatively* scheduled on **April 17 or 18, 2021** (*please reserve your time on these days*). 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 video together with the **source code** and the **project report** before the due date.

The second part is to have another **video** (**3 to 5 minutes**) to show the execution of your application based on the dataset that provided by us. A set of **documents of the running results** (*which 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. In addition, you should let us know what error(s) you have encountered and how you fix the problem.

****Note: If the size of the video is too large (cannot upload it 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 propose and implement your own scheduling algorithm that performs better than AT LEAST ANY ONE of the well-known scheduling algorithms. In addition, proper use of the two system calls, **`pipe()`** and **`fork()`**, may have a certain proportion in the marking.

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

Submission

You need to 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_Gxx.txt`**" where "**`Gxx`**" is your Group number assigned.
2. Source code and output files
   - Name of the source code file should be "**`RBM_Gxx.c`**".
   - Name the output file, as mentioned, "**`RBM_Report_Gxx.txt`**".
3. Name the project report as "**`Project_Report_Gxx.docx`**".
4. Name the batch file as "**`test_data_Gxx.dat`**" (*if applicable*).
5. Name the demo video file as "**`demo_Gxx.mp4`**"
   (*The format of the video should be in MP4. If the file size of the video is too large, send us a link to download it.*)