[2024-25] COMP2432 Operating Systems **Group Project I**
The Hong Kong Polytechnic University (PolyU) **Instructor – MOHAMMED, Aquil Mirza**
Released Date – 26 January 2025 (Sunday) **Submission Deadline** – 31 March 2025 (Monday)

SmartPark: A Smart Parking Management System for PolyU Using OS-Level Process Coordination

Scenario: PolyU University is an educational institution with a diverse population of students, faculty, and staff who require convenient access to parking facilities on campus. PolyU finds that there is a problem with its current parking management system designed 10 years ago. It is not flexible enough to make good utilization of the parking spaces and to satisfy most requests from its members. At PolyU, there are just 10 parking spaces available. Additionally, six essential facilities are offered in pairs to improve the overall user experience: (battery + cable), (lockers + umbrella), and (inflation service + valet parking). When a user reserves a battery, they automatically receive a cable as well. If a user reserves both a battery and a locker, they will be provided with a battery, cable, locker, and umbrella.

Currently, members of PolyU may simply make a reservation for a parking space or reserve only the parking facilities (essentials). For example, Member_A makes a reservation for parking space for their smart vehicle with a battery and charging cable. If they are both (parking + essentials) available, the reservation request is accepted. Otherwise, the reservation request is simply rejected. For instance, all the electric vehicle charging cable might have been booked by other members, so the request is rejected. In other words, the current system does not provide any alternative plan to satisfy or reschedule the reservation when some of the requested items or locations are not available for the requested time slot. Knowing that you have learned different scheduling methods from Operating Systems, the CFSO in PolyU is offering you a part-time job that you could use to satisfy your WIE. Would you mind helping CFSO at PolyU to revise its parking management system to achieve better utilization with an improved scheduling/rescheduling method? The goal is to improve the reservation situation so as to increase the PolyU's revenue from renting these spaces to its members.

*** *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.* ***

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 **Smart Parking Management System (SPMS)**. The project consists of the following parts:

Part I	Develop a program that allows users to add details of a booking (*name, parking slot number, date, time, duration, and/or calless etc.*) to the schedule. Besides the standard line by line input approach, SPMS 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 "*client*" or the "*requester*" and those others involved in the booking are called "*staff*". This part of the program module is referred to as the **Input Module**.
Part II	Extend the program developed in *Part I* with a scheduler to generate timetables for all bookings (e.g. available parking spots, and members booking/reservation 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 III	Augment the program with the facilities to print booking schedule for available parking slots and related additional facilities and equipment in *Part II*. Those rejected bookings should all be included. This constitutes the **Output Module**.
Part IV	Provide the program with the ability to generate a summary report to analyze the results produced in *Part III*. Compare the different schedules (*generated from different algorithms*) and find out which one would make the best use of the three parking slots. Your program should preferably be able to process N parking slots ($N = 3$ in the existing case) to cope with the expansion of PolyU 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 V	Augment the program SPMS to check whether or not the required facilities are available. If so, it assigns a priority on each booking. For example, if a parking is booked for a car and it should be equipped with a battery and charging cables, the battery and cables will be reserved for that booking with a higher priority. Even though the battery and charging cables have been reserved for another booking but without using a parking, 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.

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.

```
./SPMS
~~ WELCOME TO PolyU ~~
Please enter booking:
addParking –member_A 2025-05-10 09:00 2.0 8;
-> [Pending]
Please enter booking:
addReservation –member_B 2025-05-14 08:00 3.0 12 battery cable;
-> [Pending]
Please enter booking:
bookEssentials –member_B 2025-05-15 13:00 2.0 battery;
-> [Pending]
…
Please enter booking:
addEvent –member_E 2025-05-16 14:00 2.0 15 locker umberalla;
-> [Pending]
Please enter booking:
importBatch –batch001.dat;
-> [Pending]
…
Please enter booking:
bookEssentials –member_C 2025-05-011 13:00 2.0 battery;
-> [Pending]
…
Please enter booking:
printBookings –fcfs;
-> [Done!]
Please enter booking:
printBookings –prio;
-> [Done!]
…
importBatch –batch099.dat;
-> [Pending]
…
Please enter booking:
addParking –member_A 2025-05-16 10:00 3.0 9 battery cable;
-> [Done!]
…
…
Please enter booking:
printBookings -ALL;
-> [Done!]
Please enter booking:
endProgram;
-> Bye!
```

Some other requests would be input to 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	**./SPMS**
Use	Simply to enter [**./SPMS**] to start the program.
Command	**addParking** 如果parking，可以選擇是否加essential facilities
	addParking -aaa YYYY-MM-DD hh:mm n.n bbb ccc; **e.g. addParking –member_A 2025-05-16 10:00 3.0 battery;**
Use	It is to add a booking for a parking place with certain essentials devices together. As in the example, [**member_A**] is to make a booking on [**May 16, 2025**] at [**10:00**], and the duration is [**3.0**] hours. In addition, it also requires a battery (**battery**) and a charging cable (**cable**). [-aaa] – Member (of PolyU) 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 parking in hours (*one decimal place*) [**bbb**] and [**ccc**] – Additional essentials to be required. (*Optional, but should be in pair when request, [battery]+[cables] or [locker]+[umbrella] or [InflationService] + [valetPark]*) **;** – End of current input. *Note*: SPMS would assign a parking space which fits the requirements. Of course, SPMS would also check the availability of the required essentials (*battery and cables*). If all these are available, the request would be accepted. For [**addParking**], it is an optional choice to book any add-on essentials facilities. PolyU Members may simply book a parking space only.
Command	**addReservation**
Syntax	**addReservation -aaa YYYY-MM-DD hh:mm n.n p bbb ccc;** **e.g. addReservation –member_B 2025-05-14 08:00 3.0 battery cable**
Use	Similar to [**addParking**], it is to make a booking for a reservation. As in the example, it is to add a request made by [**member_B**] who would park his smart vehicle on **[May 14, 2025]** at **[08:00]** for **[3.0]** hours long. And, there would be 2 essentials required by the member. In addition, the parking available and should be equipped with [**battery**] and [**cable**]. [**bbb**] and [**ccc**] – Additional essentials are required (*mandatorily include*). The use of parameters is same as the previous one.
Command	**addEvent**
Syntax	**addEvent -aaa YYYY-MM-DD hh:mm n.n bbb ccc ddd;** **e.g. addEvent –member_E 2025-05-16 14:00 2.0 locker umbrella valetpark;**

Use	It is same as [**addReservation**] but it would have a higher priority if you are to use the algorithm "priority". As in the example, [**Member_E**] is to add a booking for a event on **[May 16, 2025]** at **[14:00]** for **[2.0]** hours long. The parking slot should be equipped with [**umbrella**], and [**valetpark**]. The use of parameters is same as the previous one.
Command	**bookEssentials** 用於book特定物品，表示無需捆綁銷售
Syntax	**bookEssentials -aaa YYYY-MM-DD hh:mm n.n bbb** **e.g. bookEssentials –member_C 2025-05-011 13:00 4.0 battery;**
Use	It is simply to reserve a specific essential only. As in the example, it is to reserve for **[battery_100kWh]** on **[May 11, 2025]** at **[13:00]** for **[4.0]** hours long. That request is made by [**member_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 In the example, the "FCFS" 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 "event" can replace an occupied time slot which has been assigned to a "Parking". In this context, "event" pertains to making a reservation for an occasion, such as a conference, while "parking" refers to securing a spot for parking, for instance, during 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 PolySPMS. In the other words, 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 SPMS are limited to the following:

   - five members – **`member_A`**, **`member_B`**, **`member_C`**, **`member_D`** and **`member_E`**;
   - three lockers
   - three umbrellas
   - three batteries
   - three cables
   - three valet parking
   - three inflation services

   All these could be represented by [**`child`**] processes. The parent process represents the PolySPMS 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 "*Event*" has the highest priority because it contributes the most profit. Then, it is "*Reservation*". The third one is "*Parking*". Lastly, it is "*Essentials*". In other words, a "*Reservation*" may "*displace*" an already scheduled "*Parking*" so that the caller/callee(s) involved would meet the "*Reservation*" but the "*Parking*" would be **canceled**. That could turn a previous "**Accepted**" status (*as the Parking is successfully scheduled early on*) to a "**Rejected**" one (*overtaken by this more important "Reservation" 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 booking schedule for a period, from *10 to 16 May, 2025*. Priority will be given to the booking slots between 08:00 AM – 08:00 PM. One time slot is an hour of time. That means parking 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 “booking schedule” – “ACCEPTED” and “REJECTED” may look like the following.

```
*** Parking Booking – ACCEPTED / FCFS ***

Member_A has the following bookings:

Date          Start   End     Type          Device
=============================================================================
2025-05-10    09:00   15:00   Parking       *
2025-05-12    12:00   16:00   Reservation   Battery
                                            cable
2025-05-15    09:00   12:00   Event         locker
                                            Umbrella
…
…
Member_B has the following bookings:

Date          Start   End     Type          Device
=============================================================================
2025-05-14    08:00   11:00   Reservation   Battery
                                            cable
2025-05-15    12:00   16:00   Reservation   Battery
                                            cable
2025-05-15    13:00   15:00   *             Battery
2025-05-16    10:00   15:00   Event         locker
                                            Umbrella
…
…
   - End -

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

```
*** Parking Booking – REJECTED / FCFS ***

Member_A (there are 999 bookings rejected):

Date          Start   End     Type          Essentials
=============================================================================
2025-05-16    09:00   12:00   Parking        -
2025-05-17    18:00   21:00   Reservation    Battery
                                             cable
…
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
…
Member_B (there are 999 bookings rejected):

Date          Start   End     Type          Essentials
=============================================================================
2025-05-13    13:00   15:00   Event         locker
                                            Umberalla
…
…

   - End –

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

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

```
*** Parking 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:
   …
               locker    – 99.9%
   …
               Battery   – 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:

   …
               locker  – 99.9%
   …
               Battery – 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 essentials, SPMS 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 (12 – 15 pages) that consists of the following parts:

1. [150 – 250 Words] Abstract
2. [400 – 600 Words] Introduction (*Why do you have this project*?)
3. [400 – 600 Words] Scope/Related work (*What operating systems topics have been covered in this project*?)
4. [300 – 500 Words] Concept (*What are the algorithms behind*?)
5. [300 – 500 Words] Your own scheduling algorithm (*if any*)
6. [300 – 500 Words] Software structure of your system
7. [400 – 600 Words] Testing cases/Assumptions (*How to test the correctness of your program*?)
8. [400 – 600 Words] Performance analysis (*Discuss and analyze the performance of each schedulingalgorithm you have implemented, and this could provide inputs when you design yourown algorithms*)
9. [300 – 500 Words] Program set up and execution (*How to compile and execute your project? Whichspecial libraries have been included and used in the application? For what reason theapplication needs those libraries? On which Linux server the application has beentested and what are the results*?)
10. [400 – 600 Words] Results/graphs/figures discussion
11. [200 – 300 Words] Conclusion
12. Appendix – source code file(s) and sample outputs of the application.

Note: Report template is uploaded on BlackBoard

Demonstration

The demonstration is tentatively scheduled on **Sat, April 05, 2025** (*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 the application in brief (*about 3 to 5 minutes max*) and submit it 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 the application based on data that we provide. A newer set of **reports of the running results** 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 your application within a grace period. The 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 and you 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: 30 %
Demonstration: 10 %
Bonus: 10 Points

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 and **an additional score** will be allocated if you submit the final report in a neat "formatted" style, proper references, detailed explanations, graphs/tables/flow charts (with visible headings and captions) and with a unique project title.

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

Submission

You need to submit the following files (except item 5) in ONE zipped file and upload it to the 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 "**SPMS_Gxx.c**.".
   - Name the output file, as mentioned, "**SPMS_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.*)

Group Registration: *(Deadline: Monday, February 10, 2025):* https://shorturl.at/dQ9or

Sample Documentation Report: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9830737

The above report is only for reference. You can also follow your own word count and include additional sub-sections (if required). *Report template is uploaded on BlackBoard*

NOTE:

1. Copying a project source code from friends or any online sources (including ChatAI, Chat GPT, etc.) will result in a score of ZERO points without the possibility of appeal. However, you are encouraged to use online tools for your report for checking the English grammar, rephrasing of the sentences, etc.
2. The project deadlines are set and will not be extended under any circumstances. Start early!
3. A penalty of 10% will be applied for each day the project is late.

***** The End *****