COMP 2432 Group Project (2019/2020 Semester 2)

Submission deadline: 8 May 2020

Weighting: 18%

Project title: **Production Line Scheduler** (**PLS**)

Scenario

PolyComp is a medium-size manufacturer and owns three plants to produce a number of different products. On average, the three plants (*Plant_X, Plant_Y and Plant_Z*) can produce 300, 400 and 500 products a day respectively (*currently, they are producing surgical mask, respirator, and medical protective clothing*). Recently, the factory manager found that there were a number of orders which could not be completed on schedule and caused the decline of the company's profit. After an investigation, it was found that the three plants were not fully utilized because there was no good planning of the production schedule for the three plants, causing the utilization of the plants to become low. Knowing that you have just learnt the algorithms of "scheduling" in Computing, PolyU, the factory manager would like to seek help from you to advise them how to make a better way to schedule the production.

In this project, you are asked to create an application that can help the company to schedule their production in order to produce the best utilization of the three plants. In addition, the application should also report which order should be accepted and which one should be rejected. This is because if an order was accepted but could not be completed on time that would cause a loss not only in short-term profit, but also in long-term goodwill.

Project Requirements

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

Part 1. Write a program that allows user to add details of orders (due date, quantity, product name, etc) to the scheduler. This is referred to as the **Input Module**.

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

Part 3. Augment the program with the facilities to print out schedules for the factory manager 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 to analyze the results produced in *Part 3*. Compare the different schedules (generated from different algorithms) and find out which one would be the best use of the three plants. Your program should preferably be able to process *N* plants (*N* = 3 in the existing case). By the way, an outstanding (*rejected*) list may be included in this report for those orders cannot be scheduled. This final module is the **Analyzer Module**.

You must form a group of 3 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 Linux Servers** (*apollo* or *apollo2*) in the Department of Computing. You will have to specify to us on which Linux server your project has been tested and/or executed.

***** *Note that "`cc`" or "`gcc`" is the compiler that we would use in this project.* *****

Implementation

User Interface

A user interface is needed for user to input commands and to add/create orders and activities in the product line scheduler application (**PLS**). The input methods may look like the followings (*blue colored lines*).

```
   ~~WELCOME TO PLS~~

Please enter:
> addPEIOD 2020-06-01 2020-06-30
Please enter:
> addORDER P0001 2020-06-10 2000 Product_A
Please enter:
> addORDER P0002 2020-06-13 3000 Product_D
Please enter:
> runPLS FCFS | printREPORT > report_01_FCFS.txt
Please enter:
> addBATCH orderBATCH01.dat
Please enter:
> runPLS PR | printREPORT > report_02_PR.txt
> addORDER P0202 2020-06-23 3000 Product_H
…
…
…

Please enter:
> runPLS FCFS | printREPORT > report_99_FCFS.txt
Please enter:
> exitPLS
Bye-bye!
```

Input Format

For the inputs, most likely they all are in strings and we assume there are no errors for those input contents/values.

No matter whether inputs are provided line by line through the interface of the program or imported through a batch file in plain text format, you are recommended to input a large number of requests which ***exceed*** the capacity of the available production capacity in your testing process so as to produce an outstanding list. You are also strongly recommended to test for fewer requests to check whether or not the algorithms implemented the work as expected. In other words, it is very important to thoroughly test your program with various types of inputs, inclusive of those hitting boundary conditions.

Input	`addPEIOD 2020-06-01 2020-06-30`
Format	**`addPERIOD`** `[start date] [end date]`
Usage	It is to specify the period, (start date and end date) for scheduling the production. Date format is year-month-day, i.e. YYYY-MM-DD.

Input	`addORDER P0001 2020-06-10 2000 Product_A`
Format	**`addORDER`** `[Order Number] [Due Date] [Quantity] [Product Name]`
Usage	[addORDER] is to add an order and the details to the scheduler. Follow by the Order Number, Due Date, Quantity and Product Name. In this project, we assume there are 9 products in 3 categories. Product_A, B and C belong to Category_1; Product_D, E and F belong to Category_2; and Product_G, H and I belong to Category_3. If "Priority" is applied in your algorithm, Category_1 has the highest priority and then Category_2 and the lowest one is Category_3.

Input	`addBATCH orderBATCH01.dat`
Format	**`addBATCH`** `[Orders in a batch file]`
Usage	[addBATCH] is to input multiple orders in one batch file which is a basic text format. That is there are many lines of "addORDER" in the file.

Input	`runPLS FCFS | printREPORT > report_01_FCFS.txt`
	`runPLS PR | printREPORT > report_02_PR.txt`

Format	**`runPLS`** `[Algorithm]` **`|`** **`printREPORT`** `> [Report file name]`
Usage	[runPLS] is to generate a schedule with the specified [Algorithm]. is to execute a scheduling algorithm. By using the vertical bar [|], the schedule is passed to [printREPORT] command and to print a report of that schedule with the analysis details, for example, the utilization. The greater than sign [>] is to export the report to the given file name. The file name is composed by "report_" + [a sequence number] + [algorithm used].

	Algorithm might be used: - FCFS: First Come First Served - PR: Priority - SJF: Shortest Job First - ??? *** You may have your own algorithm to schedule the production.

Input	exitPLS
Usage	[exitPLS] is to terminate the program. Here we are expecting the program is terminated properly. For example, all child processes have been terminated and collected successfully.

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

1. One time slot is assumed as one day. The plants cannot be changed to produce another product in the middle of the day without being "reconfigured" and "booted" at the beginning of every day at midnight. For example, if there is an order of 1000 items of Product_A, the scheduler may produce any one of the following arrangements:
    a. Plant_X is used for 4 days (1000/300 = 3.33 days, i.e. 4 days)
    b. Plant_Y is used for 3 days (1000/400 = 2.5 days, i.e. 3 days)
    c. Plant_Z is used for 2 days (1000/500 = 2 days)
    d. Any combination of the usage of the plants, for example, Plant_X + Plant_Y may have the following:
        i. Plant_X is used for 2 days (300 x 2 = 600) + Plant_Y is used for 1 day (400)
        ii. Plant_X is used for 1 day (200) + Plant_Y is used 2 days (400 x 2 = 800)
        iii. Other possible combinations to produce 1000 items of Product_A.
2. The first command of the PLS is [addPERIOD] because it allows the application to know how many days would be concerned in generating the schedule.
3. There is no rest day in the period. In the other words, there is no need to consider holiday. The three plants work 24x7, except for potential maintenance and reconfiguration operations at every mid-night.
4. All inputs are stored in a separate file as the raw data in the application.
5. A schedule is generated until a command is called. And, according to which algorithm is invoked, the raw data would be processed at that moment of call.
6. At least there are TWO different algorithms implemented in the application.
7. 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. Similarly, the output module can 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 "**`|`**").

8. If **`pipe()`** and **`fork()`** system calls are both used properly in the program, the maximum possible mark is 100%. On the other hand, if both system calls are not used in the program, only a passing mark (50% to 55%) would be awarded. Intermediate scoring will be given if only one system call is used, depending on the extent of usage.
9. Bonus score of 5% would be given if you can propose a better algorithm which is not the one of the existing algorithms taught in the class.

Output Format

The format of the production schedule 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 schedule.

Plant_X (300 per day)
2020-06-01 to 2020-06-30

Date	**Product Name**	**Order Number**	**Quantity (Produced)**	**Due Date**
2020-06-01	Product_A	P0001	300	2020-06-10
2020-06-02	Product_A	P0001	300	2020-06-10
>>> >>>	…	…	…	…
2020-06-08	…	…	…	…
2020-06-09	NA			
>>> >>>	…	…	…	…
2020-06-18	…	…	…	…
2020-06-19	…	…	…	…
2020-06-20	…	…	…	…
>>> >>>	…	…	…	…
2020-06-29	…	…	…	…
2020-06-30	…	…	…	…

The table simply indicates the "date", "products" and the "quantity produced". Here "NA" means the timeslot is NOT scheduled.

In addition, the analysis report may look like the following one.

```
***PLS Schedule Analysis Report***

Algorithm used: XXXXXXXXXXXXXXXXXX

There are XX Orders ACCEPTED.  Details are as follows:

ORDER NUMBER   START         END             DAYS     QUANTITY     PLANT
===========================================================================
P0001          2020-06-01    2020-06-02         2         600      Plant_X
P0001          2020-06-01    2020-06-01         1         400      Plant_Y
P0002          2020-06-DD    2020-06-DD        XX        XXXX      ???
… … …
… … …
… … …

                - End -

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


There are XX Orders REJECTED.  Details are as follows:

ORDER NUMBER   PRODUCT NAME     Due Date        QUANTITY
===========================================================================
P00XX          Product_?        2020-06-DD         ????
P00XX          Product_?        2020-06-DD         ????
… … …
… … …
… … …

                - End -


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


***PERFORMANCE

Plant_X:
         Number of days in use:                     ?? days
         Number of products produced:            ????? (in total)
         Utilization of the plant:                ??.? %

Plant_Y:
         Number of days in use:                     ?? days
         Number of products produced:            ????? (in total)
         Utilization of the plant:                ??.? %

Plant_Z:
         Number of days in use:                     ?? days
         Number of products produced:            ????? (in total)
         Utilization of the plant:                ??.? %

Overall of utilization:                           ??.? %
```

In this part, it is simply to list out all the orders of which are accepted or rejected. The requested products are manufactured in which plant and how many were produced.

The utilization is computed in this way:

$$\frac{\textit{Total Amount of Product Produced}}{\textit{Total Quantity Could Be Produced}}$$

For example, Plant_X can produce 300 products a day. In 10 days, it can produce 3000 pieces. However, if Plant_X has just produced 2700 in ten days. The utilization of Plant_X is 2700/3000 = 90%.

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 is out of the "period range" entered, correction should be taken place automatically. For example, you may treat that entry as "invalid" and store it in a separate file.

Documentation

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

1. Introduction (*Why do you have 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. Software structure of your system
6. Testing cases (*How to test the correctness of your program*?)
7. Performance analysis (*Discuss and analyze the performance of each scheduling algorithm you have implemented, and this could provide inputs when you design your own algorithms*)
8. Program set up and execution (*How to compile and execute your project? Which special libraries have been included and used in the application? For what reason the application needs those libraries? On which Linux server the application has been tested and what are the results*?)
9. Appendix – source code file(s) and sample outputs of the application.

Demonstration

The date of demonstration is tentatively scheduled on **May 9 or 10, 2020** (*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*) 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.

Mark distribution

The mark distribution of this project is as follows:

Implementation:	60 %
Documentation:	25 %
Demonstration:	15 %
Bonus:	5 %

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 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_Gxx.txt`**" where "**`Gxx`**" is your Group number assigned.

2. Source code and output files

   - Name of the source code file should be "**`PLS_Gxx.c`**.".
   - Name the output file, as mentioned, "**`PLS_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`**" (*Use MP4 format for your video. If the file size of the video is too large, send us a link to download it.*)