COMP2432
G06 Group Project Reoprt

Room Booking Manager

19081789D MAN, Furui
19078543D WANG, Meng
18080998D WU, Junyu
19079008D XING, Shiji

Contents

1 Introdoction

The project aims to utilize the knowledge covered in COMP2432 Operating Systems and put them into practice to get a further understanding and improvement. This works out by implementing a facility management system for a company, which would make advantage of all kinds of scheduling skills and abstractions covered in the lecture, along with the use of multi-process programming and inter-process communication.

2 Scope

Multi-Process Programming and Inter-process Communication: pipe() and fork();

CPU Scheduling: FCFS and PR

Memory Allocation: MFT

Synchronization: Program-Monitored Synchronization

3 Concept

3.1 FCFS Scheduling

First-Come-First-Serve(aka, FCFS) handles requests upon a first-come-first-serve basis. Later requests that cause collision are rejected, otherwise are accepted.

In its implementation, no sorting is needed because the order of the request link-list during scheduling is exactly the same as that of input, given that the invalid ones have been filtered out. The only thing to note is that to maintain the order of requests, Input-Handler Module must be single-threaded.

Below is a illustration of FCFS algorithm. Green-colored requests are accepted, whereas red-colored requests are rejected, and the requests arrive in sequence indicated by arrows between.

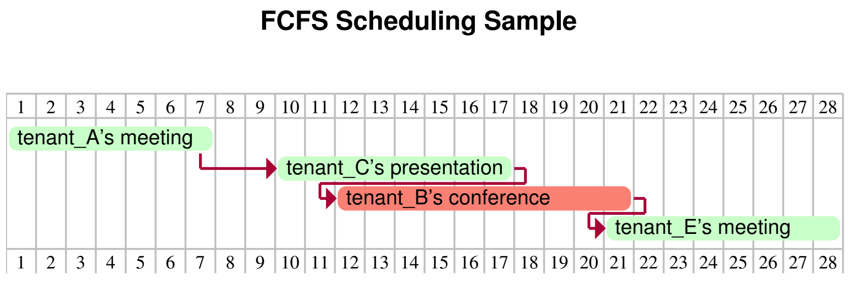

Figure 1: Gantt Diagram for FCFS algorithm Sample output for one single Room

3.2 PRIO Scheduling

Priority scheduling(aka, PRIO) is the scheduling algorithm based on the priority of requests. Rather than that in FCFS which indicated by arriving time, the priority are implied within the requests weighted by its type (i.e. device-booking, meeting, presentation, or conference).

To implement PRIO, the tasks required is to stably sort the request link-list upon user-defined priority, then call FCFS scheduling. This would reuse the code and decrease the overall complexity of the program. Requests are stored in an array after sortion based on priority.

Below is the visualization of PRIO scheduling, with the same annotation rules as used in FCFS.

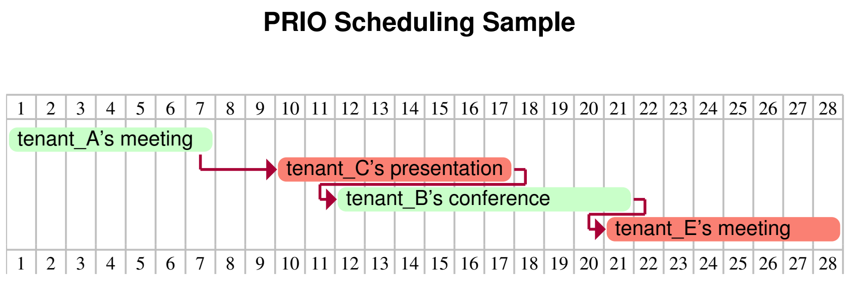

Figure 2: Gantt Diagram for PRIO algorithm Sample output for one single Room

4 Algorithm

4.1 Design of own algorithm

Optimization Algorithm

Optimization algorithm is based on processed result by FCFS algorithm or PRIO algorithm. Failed requests from the two algorithm firstly undergo verification. Valid requests are rescheduled based on bi-directional search of linked lists of rooms and devices.

5 Program Structure

5.1 Class Design

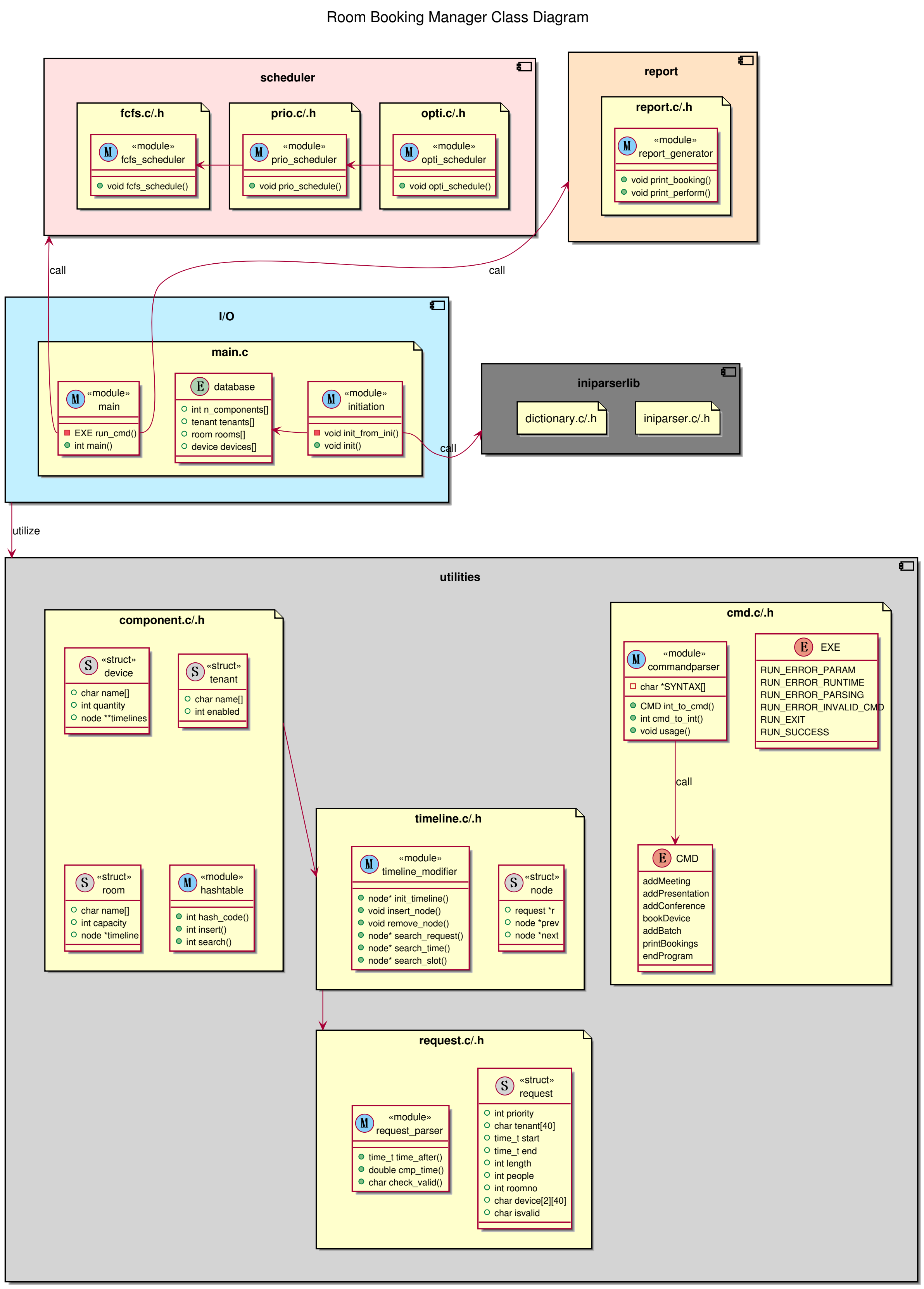

Figure 3: Overall class design diagram of Room Booking Manager

5.2 Sequence Design

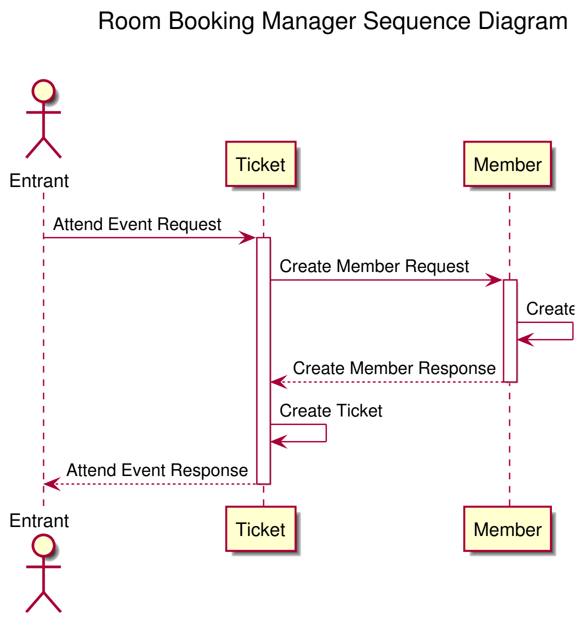

Figure 4: Overall sequence design diagram of Room Booking Manager

5.3 Activity Design

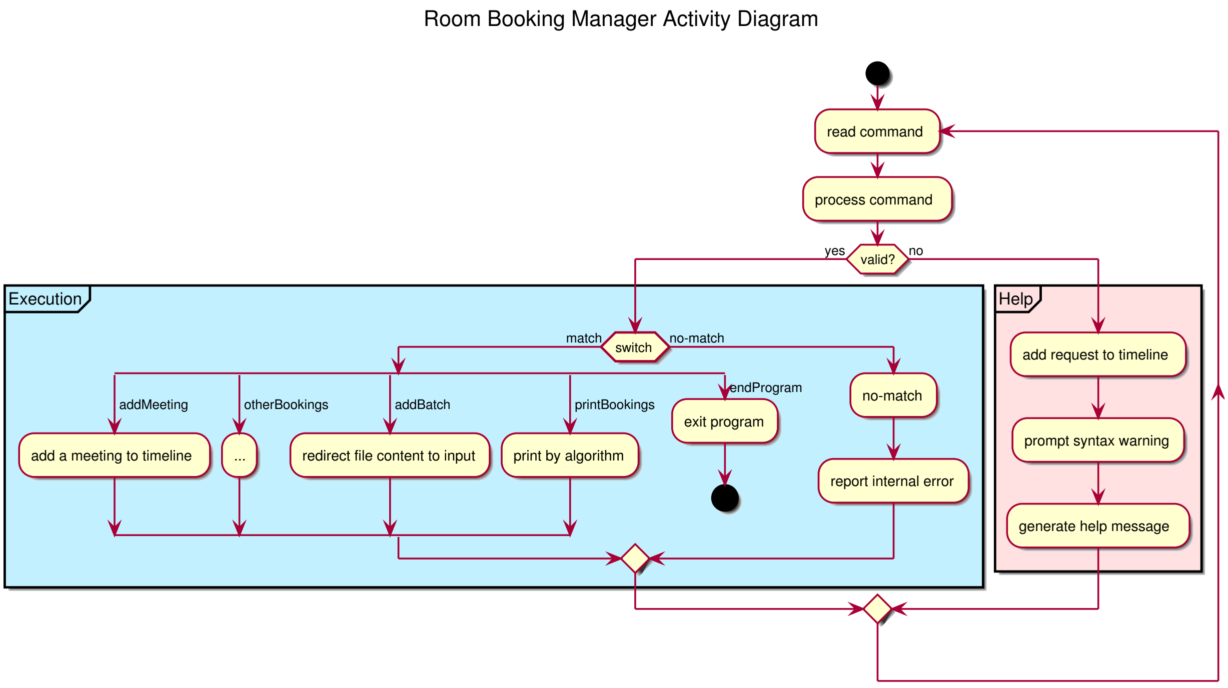

Figure 5: Overall activity design diagram of Room Booking Manager

6 Testing Cases

This is the brief version, which demonstrates the valid and invalid tests for `addMeeting` instruction only.

Tests for other instructions are similar and therefore not included. Syntax of other instructions varies in number of the parameters. Help message on syntax is available once an input error is detected.

Refer to the appendix for the full version.

6.1 Valid Tests

Valid syntax for `addMeeting` instruction should be:

```
addMeeting -tenant YYYY-MM-DD hh:mm n.n [d1 d2];
```

Below are some samples which conforms the above syntax:

```
addMeeting -tenant_A 2021-05-10 21:50 1.50 5 projector_2K screen_100;
addMeeting -tenant_A 2021-05-11 18:20 0.30 5;
addMeeting -tenant_A 2021-05-11 4:10 0.0 5;
```

6.2 Invalid Tests

Invalid instructions of addMeeting contains either:

- Syntax invalid: (including command invalid, tenant invalid, date invalid, hour and minute invalid, duration invalid, number of people invalid, device invalid); or

```
command_invalid and parameters does not matter;
addMeeting -tenant_invalid 2021-05-10 1:30 0.50 5 projector_2K screen_100;
addMeeting -tenant_A date-in-valid 1:30 0.50 5 projector_2K screen_100;
addMeeting -tenant_B 2021-05-10 hhmm:invalid 1.50 5 webcam_FHD monitor_75;
addMeeting -tenant_C 2021-05-10 18:30 duration.invalid 5 webcam_FHD monitor_50;
addMeeting -tenant_D 2021-05-10 10:40 0.0 peopleinvalid projector_2K screen_150;
addMeeting -tenant_D 2021-05-16 3:10 1.10 5 device_invalid monitor_50;
```

- Device pairing error (devices must be in pairs).

```
addMeeting -tenant_E 2021-05-10 22:20 0.50 5 projector_4K monitor_50;
addMeeting -tenant_E 2021-05-10 22:20 0.50 5 projector_4K;
```

7 Performance analysis

8 Program Setup & Analysis

8.1 Program Setup

Step 0 Clone repo (optional)

Clone the repo from Github if there is no local copy.

```
git clone https://github.com/toolsmax0/COMP2432_RBM.git
```

Step 1 Compilation

`cd` to the project's root directory and execute `build.sh` script.

The program have dependency upon `gcc 4.0+` and `linux 3.0+`.

```
cd COMP2432_RBM
sh build.sh
```

Step 2 Customization (optional)

To modify the component settings (i.e. tenants, rooms, devices), modify `RBM.ini` file according to its syntax.

Step 3 Execution

To execute the program, run the following command.

```
./out/RBM
```

8.2 Progarm Analysis

9 Appendix

9.1 Source Code

All source files are located under `./src/` directory. Please `cd` to corresponding directory for reference.

```
Insert source here
```

9.2 Test Data

All test data are generated with `generator.py` under `./test/` directory. Sufficient amount of test cases are generated via this generator and stored into `*.dat` files.

All files of test data are located under `./test/` directory. Please `cd` to corresponding directory for reference.

Files marked with `*_invalid.dat` are invalid tests for specific command types. The rest of files are for valid tests.

```
Insert test data here
```

9.3 Test Results

Warnings should be generated for each invalid case, while not for valid cases.

The descriptions of valid/invalid tests have been clearly stated under **9.2 Test Data**.

```
Insert warning here
```

9.4 Sample Output

Output of test files are stored in `sample_output_*.txt` file under `./test/` directory with algorithm applied. Please refer to the corresponding file for reference.

```
Insert Output file here
```