****

**COMP2432 Operating Systems**

**Group Project (2015/2016 Semester 2)**

**Room Booking Mana**g**er (RBM)**

Group 14

|  |  |
| --- | --- |
| CHEN Yunkun | 13103214d |
| LIU Chang | 13107851d |
| ZHENG Hongyi | 13104036d |

1. **Introduction**

In the scenario that the PolySME’s room booking system has a design problem, the system is not flexible enough to satisfy most requests and cannot make good utilization of the meeting rooms. One request from the user may include booking for rooms and some facilities. The current problem is that if any one item in the request is not available, the whole request will be rejected. The system will not provide any alternative plan that the client may fail to book the room and facilities. In order to solve this problem and improve the performance of the current system, a new system called Room Booking Manager (RBM) is developed.

The new system can be used for its client to book meeting rooms as well as some additional facilities, which include projectors, screens, webcams etc. Considering there may be multiple requests from different clients, RBM is designed to be flexible enough to arrange the booking items for different request. Moreover, RBM is able to make alternative plan to satisfy the request in the case that the booking facilities in the request may not be entirely available. All the functions are achieved by a better scheduling algorithm on the basis of the old system, and there is no doubt that the new system can significantly increase the revenue of this company.

Based on the assumed scenario, this project offered a good opportunity to apply what have been taught in the subject of COMP2432 to the real case. During the lecture, the concept of scheduling process and some basic algorithms were introduced. When developing RBM, the scheduling algorithm was considered because the arrangement of the booking can be regarded as a scheduling problem. In order to make the system more efficient, multiple scheduling algorithms were attempted. In addition, the idea of child process and pipe have also been introduced to this project in the purpose of higher efficiency that allows different child to handle different task. Following the guidance and requirement, this project provide a good practice to deal with the problem from real case. Although the process was complex and even boring sometimes, a higher learning outcome on the concept of operating systems is achieved though working on the project.

1. **Scope**

The whole project is written in C language and will be compiled in the Linux system, and multiple processes are created to handle different booking request. Therefore, the involved topics mainly include Linux programming, process management and scheduling algorithm.

* 1. Linux Programming

C language is used in order to make the system executable on the Linux system of COMP department. To implement the requirement, the project was divided into three major parts – input, schedule and output modules. The input module is going to handle the request from user and then the schedule module will utilize proper algorithm to make arrangement. Finally, the output module will print out all the necessary information. And in order to store the information through all processes, a linked list was created that each element in the list will record one request from the user (see Figure 1).

* 1. Process Management

As there are multiple modules and the scheduler may be seen as a separate program, different processes were created to make it more efficient. To generate a child process, *fork()* function is called, thus the separating parts can work synchronously. And in order to distribute tasks to different child process, the idea of pipe is also used. The *pipe()* function is used to create a communication channel between the parent and child processes so that the parent can send order to the child process.

* 1. Scheduling Algorithm

The scheduler in this project is required to schedule the requests in a proper manner, and three different algorithms, which include First come and first serve algorithm (FCFS), Priority algorithm (PRIO) and Optimized algorithm (OPTI), are adopted for different purpose. On the basis of different algorithm, the arrangements may be different. The requests will be scheduled by all the three algorithms, and the client is able to print out each result of different approach.

1. **Concept**
   1. First come first serve algorithm

In this algorithm, if the system receives a request, the information will be first send to the child. Then the child process will check whether the requested room or facilities is available or not. If not, the request will be accepted. Simply, the program will comply with the following procedures:

1. A pointer points to the first element in the linked list and send to child process;
2. Child process first check if the booking items in the request are available;
3. If yes, this request is accepted;
4. If not, skip this element;
5. Point to next element in the list, repeat the above steps.
   1. Priority algorithm

Priority algorithm will be based on the priority of the booking purpose. In the assumption, the priority for the usage of room should be conference > presentation > meeting > device. The request with higher priority purpose will displace the bookings with lower priority. To implement this algorithm, the scheduling mechanism will start with the highest priority “3” and start the next round after traversing through the whole linked list. The processing procedures are:

1. A pointer points to the first element in the linked list with highest priority (type=3) and send to child process;
2. Child process first check if the booking items in the request are available;
3. If yes, this request is accepted;
4. If not, skip this element;
5. Point to next element in the list with same priority and repeat the above steps;
6. If the pointer traverse to the tile of the list, start from step 1 with lower priority (type=2, type=1 and type=0 respectively).

(Means of type – 3: conference; 2: presentation; 1: meeting; 0: device. Defined in header file, bigger number means higher priority.)

1. **Our Scheduling Algorithm (Genetic Algorithm + Rescheduling Algorithm)**

Our scheduling algorithm can generate better output than both First come first serve algorithm and Priority algorithm. Our scheduling algorithm is separated into two parts. Firstly, genetic algorithm is used to select a group booking from the list of booking requests. Secondly, for all rejected bookings, rescheduling algorithm is used to reallocated them into a new available time slot.

* 1. Genetic algorithm

A genetic algorithm is used to ensure the output of fist part is better or at least equal to First come first serve algorithm and Priority algorithm.

* + 1. Basic concepts

Genetic algorithm is a method for solving constrained optimization problems based on a natural selection process that mimics biological evolution. The algorithm repeatedly modifies a population of individual solutions.

* + 1. Gene and chromosome

Each gene is corresponding to one booking request. The value of the gene represent that it is accepted or rejected (1 for accepted and 0 for rejected). Assume there are tally n booking requests, then each chromosome contains n genes, which represent a selection combination of current bookings.

* + 1. First generation

First generation contains 4 chromosomes. Two of them are generated by First com first serve algorithm and other of them are generated by Priority algorithm. This can ensure that, after several generations, the output of genetic algorithm would be better than both of two algorithms.

* + 1. Fitness function

Fitness function is used to calculate the fitness of certain chromosome (certain accept-reject scheduling plan). We use the average time utilization of all rooms and devices to represent the fitness. That means, the accept-reject plan with higher average time utilization will have higher fitness value. The reason to choose this criterion is that the performance of scheduling algorithm is assessed by the average time utilization.

* + 1. Selection

In each generation, the algorithm selects the two chromosome with highest fitness as the parent of next generation.

* + 1. Crossover and mutation

The two selected chromosome crossover to generate four chromosome of next generation. The crossover point is generated by a random number. In each generation, there are 5 possible mutations with 10% possibility to occur.

* + 1. Termination

We let the genetic algorithm to run for 100 generations. Then pick the chromosome with highest fitness value. It would yield a generation which is better or at least equal to the output generated by First come first serve algorithm and Priority algorithm.

* 1. Rescheduling algorithm

After genetic algorithm generated the selection scheme, rescheduling algorithm is used to reallocate all the rejected booking.

* + 1. Sort

Firstly, all the bookings are sorted according to their average time utilization. The rejected booking with higher average time utilization will be rescheduled first to ensure the better performance.

* + 1. Reallocate

For each rejected booking, their request time will be postponed 1 hour by each step, then check whether the time slot is available. If the time slot is available, the booking will be allocated. If not, further postpone 1 hour. If there is no time slot available until the last hour, the booking time will be set to the first hour of the two working weeks and continue the check process.

1. **Software Structure**

There are totally 6 files in the project, which include a header file and five C files. The header file stores all the declarations, and each of the C files serve for a one function such as inputModule and scheduler.

5.1. Header file – *header.h*

In the header file, it first includes all the system header files that are useful in this project. Then it declares a linked list, which includes all the necessary information, to record each booking request. And when there is a booking request, a new element will be created. At the end of this file, it also declares all the functions that will be used in the program.

5.2. Main C file – RBM*.c*

After the program is executed, the program will first print out a welcome statement and initialize the variables. Afterwards, two linked lists will be created with two head pointers head and head\_s respectively. Additionally, multiple child processes are created for serving different functions. And there are two pipes (parent\_to\_child and child\_to\_parent) being created for each child to make the parent and child processes communicate. Lastly, the program will stand by and call the function *WaitForBooking()* from input module to wait for the user’s command.

5.3. Input function – *inputModule.c*

The input module is to receive the input from the user and analyze the stream for further processing. The *fscanf()* function is called to store the input into a string. And the algorithm to classify the input is to verify the fourth character of the string. The reason to adopt this method is that the fourth characters of all functions are different and the available commands are in different size in the character. For example, ‘M’ stands for “addMeeting” and ‘n’ represents “printBookings”.

Because there will be spaces in one command, so every command input will be divided into several pieces on the basis of spacing standing for different requirements. For example, if the input is “addPresentation –room\_B 2016-04-04 09:00 3.0 tenant\_B screen\_100”. It will be divided into 6 parts – “room\_B”, “2016-04-04”, “09:00”, “3.0”, “tenant\_B” and “screen\_100”. Then all these informations will be recorded into the linked list in terms of integers. The corresponding variables in the block will be assigned with value “1” if it is requested. After confirming which room and devices are requested, the input module will send the time data to scheduleModule for time arrangement.

5.4. Scheduler – *scheduleModule.c*

Three algorithms for scheduling are used to arrange the requests. Except First Come First Serve (FCFS) and Priority (PIOR) algorithms, an optimized algorithm was developed that would have a better scheduling mechanism than the others. The basic algorithms for FCFS and PIOR are introduced in the 3rd section of this report, and the optimized algorithm is introduced in the 4th section of this report.

5.5. Output function – *outputModule.c*

This module is used to print out the “accepted” and “rejected” bookings in the format shown in the Figure X. As there are three different algorithms, different information will be printed out due to different command –*printBookings –fcfs*, *printBookings -prio* and *printBookings – opti*. After the linked list is scheduled by the scheduler, the pointer will traverse the list four times in the order of:

1. Traverse from head, print requests that is accepted and room is A;
2. Traverse from head, print requests that is accepted and room is B;
3. Traverse from head, print requests that is rejected and room is A;
4. Traverse from head, print requests that is rejected and room is B;

5.6. Data structure and communication function – *helpModule.c*

This module is used to determine the data structure of the system, including a linked list to store booking records. In each record, some integers are used for storing the devices and rooms which the booking request for. Function *CreateRecordList()* is for creating the linked list, and *AddRecord()* is used for adding records into created linked list. In addition, function Send() is used for the communications between parent and child processes.

1. **Testing cases**

In order to test the correctness of the algorithms, the following booking requests are input in order:

1. addPresentation -room\_B 2016-04-04 09:00 3.0 tenant\_B projector\_fhd screen\_100
2. addConference -room\_A 2016-04-08 10:00 3.0 tenant\_A webcam\_1080p
3. addMeeting -room\_A 2016-04-10 12:00 2.0 tenant\_C
4. addPresentation -room\_A 2016-04-14 14:00 1.0 tenant\_A
5. addPresentation -room\_B 2016-04-09 12:00 2.0 tenant\_B screen\_100
6. printBookings -fcfs/prio/opti (**Check Point 1**)
7. addPresentation -room\_A 2016-04-09 12:00 2.0 tenant\_A screen\_100
8. printBookings -fcfs/prio/opti (**Check Point 2**)
9. addMeeting -room\_B 2016-04-09 11:00 2.0 tenant\_C
10. printBookings -fcfs/prio/opti (**Check Point 3**)
11. addConference -room\_B 2016-04-09 12:00 3.0 tenant\_A
12. printBookings -fcfs/prio/opti (**Check Point 4**)

Check Point 1

The results of all the algorithms based on the first five inputting should be the same because there is no time clash. **Figure 1** shows the result.

Check Point 2

A new booking (No.7) is made by another tenant that is similar a existing booking (No.5), but the two tenants booked different rooms. One problem appears that the equipment screen\_100 is required at the same time, thus the three algorithms have different outcome.

* FCFS and PRIO rejected the new booking (**Figure 2**);
* OPTI rescheduled the new booking to a new time slot that just follows the requested time (**Figure 3**).

Check Point 3

A new booking (No.9) is made that requires Room A at a time slot booked by a existing booking (No.5), and the priority of the old booking is higher than the new one.

* FCFS rejected the new booking because of time clash (**Figure 4**);
* PRIO rejected the new booking because the priority of Meeting is lower than Presentation (**Figure 4**);
* OPTI rescheduled the new booking to a new time slot that is just after the current booking (No.5) (**Figure 5**).

Check Point 4

A new booking (No.11) requires a Conference Room that has the highest priority, but the time it required still clashed with the earlier booking (No.5).

* FCFS rejected the new booking because of time clash (**Figure 6**);
* PRIO accepted the new booking because the priority of Conference is higher than Presentation. At the same time, because the booking No.5 is rejected, equipment screen\_100 became available. Therefore, the rejected booking (No.7) is rescheduled into the timetable. (**Figure 7**);
* OPTI rescheduled all the three bookings (No.5, No.9 and No.11) that request for the same time. (**Figure 8**).

1. **Performance analysis**

Totally, three algorithms are used in this system, which are First Come First Serve (FCFS), Priority (PIOR) and Optimized algorithm.

Using the test case mentioned in 6th section of this report, we can generate the performance of different algorithms. Figure 9 is the performance of FCFS, Figure 10 is the performance of PIOR, and Figure 11 is the performance of Optimized algorithm.

As shown in the figures, FCFS has the worst performance since it rejected 3 bookings, PIOR is better since it rejected 2 bookings, and Optimized algorithm is the best since it accepted all bookings. Actually, the advantage of Optimized algorithm will be more significant when the bookings become more.

1. **Program set up and execution**

There are totally 9 files in our project making up the program, which include a default batch file, a header file, a makefile and 6 C files, and they are -

* header.h
* RBM.c
* inputModule.c
* scheduleModule.c
* outputModule.c
* helpModule.c
* geneticAlgorithm.c
* tests.dat
* makefile

Because the tests that have been done to examine the project in based on the Linux system ***apollo2*** of the COMP department, it is preferred to set up and execute the program on apollo2 or apollo. And in order to compile the program, it is necessary to compile all the C files together using the system compiler *cc.* To simplify the compiling process, a makefile is made to compile all the files together by entering the command “make” (Figure X).

After the compilation is finished, the executable file RBM will be created, and another order “./RBM” is required to start the program. And a line of information “~~ WELCOME TO PolySME ~~” will be printed out after successful log in (Figure X). To test the program, the available syntaxes include the following –

1. addMeeting –aaa YYYY-MM-DD hh:mm n.n bbb

e.g. addMeeting –room\_A 2016-04-05 09:00 2.0 tenant\_A

1. addPresentation –aaa YYYY-MM-DD hh:mm n.n bbb ccc ddd

e.g. addPresentation –room\_B 2016-04-04 09:00 3.0 tenant\_B screen\_100

1. addConference –aaa YYYY-MM-DD hh:mm n.n bbb ccc ddd

e.g. addConference –room\_A 2016-04-06 14:00 2.0 tenant A webcam\_720p

1. addDevice –aaa YYYY-MM-DD hh:mm n.n bbb;

e.g. addDevice –projector\_xga 2016-04-07 13:00 2.0 tenant\_b;

1. addBatch –xxxxx

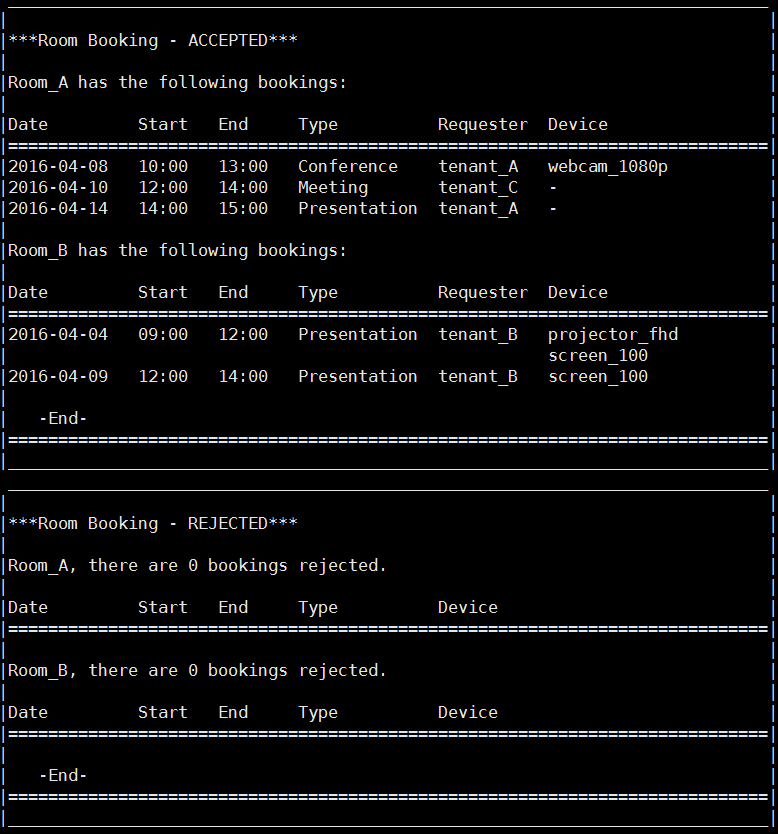
e.g. addBatch –batch001.dat

1. printBookings –xxx -[fcfs/prio/opti]

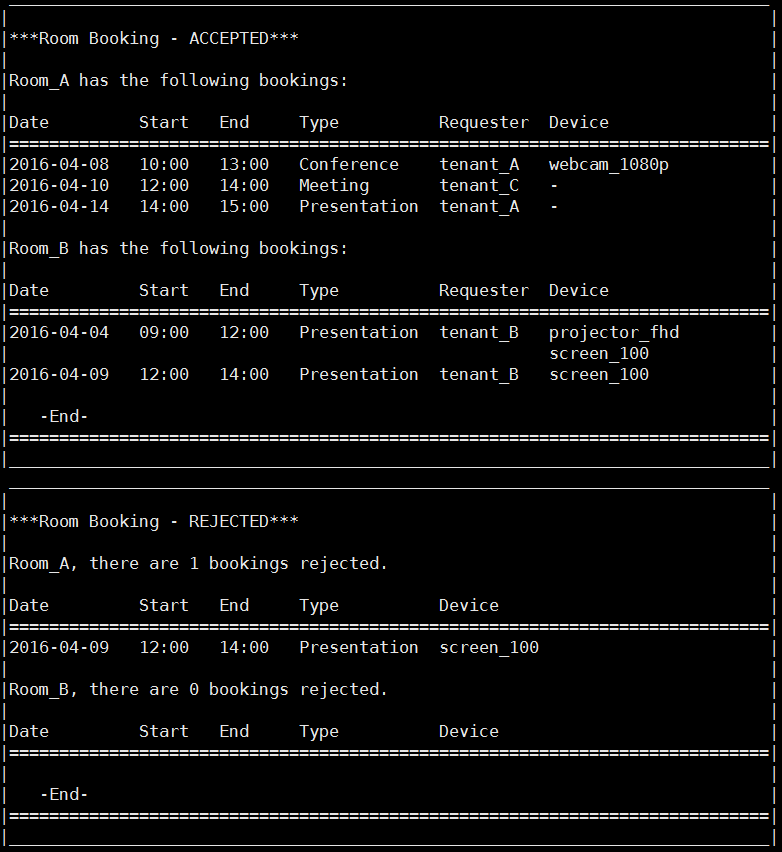
e.g. printBookings -fcfs

1. endProgram
2. **Appendix**

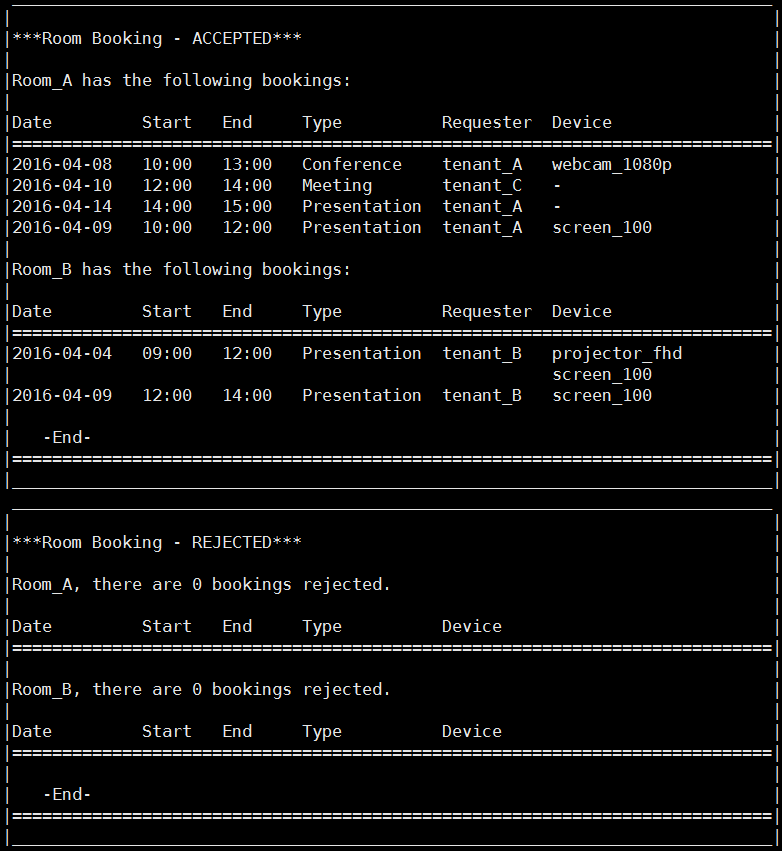
**Figure 1**

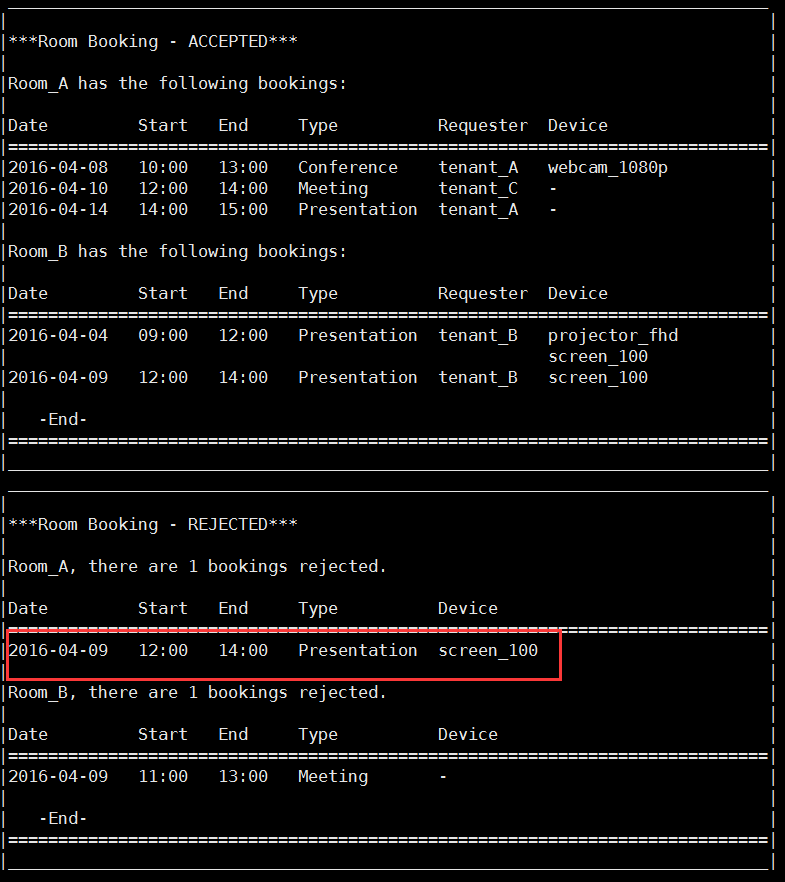


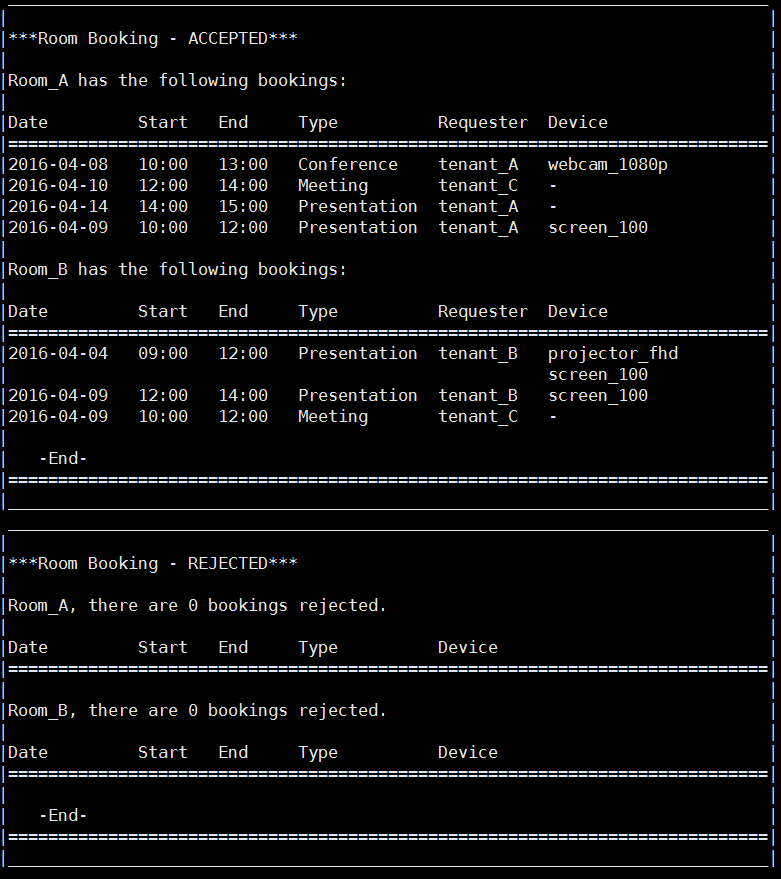
**Figure 2**

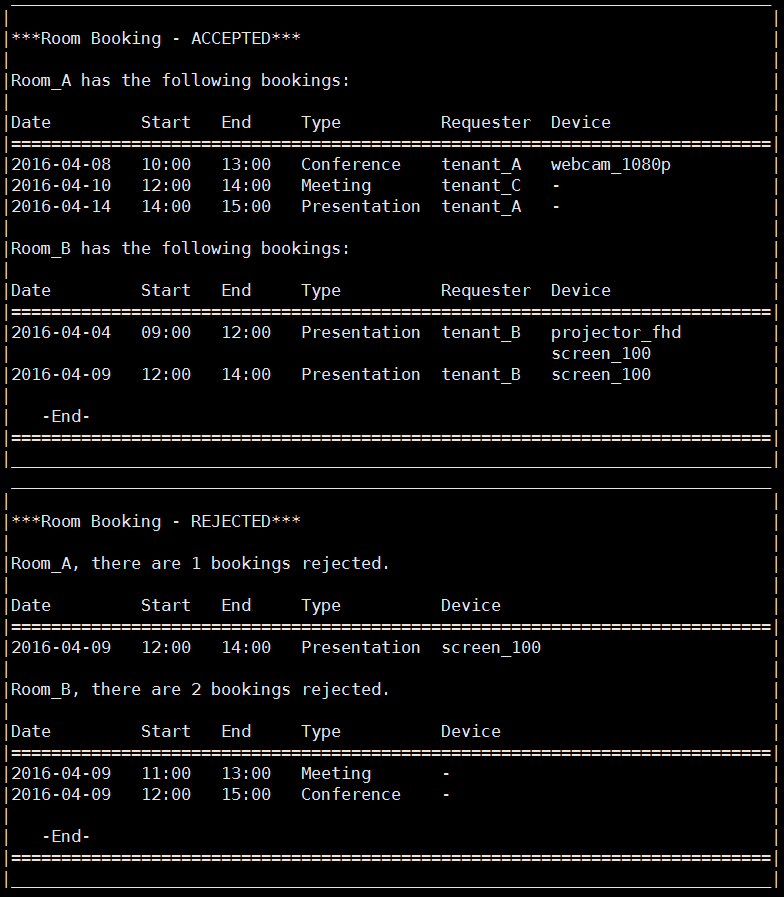


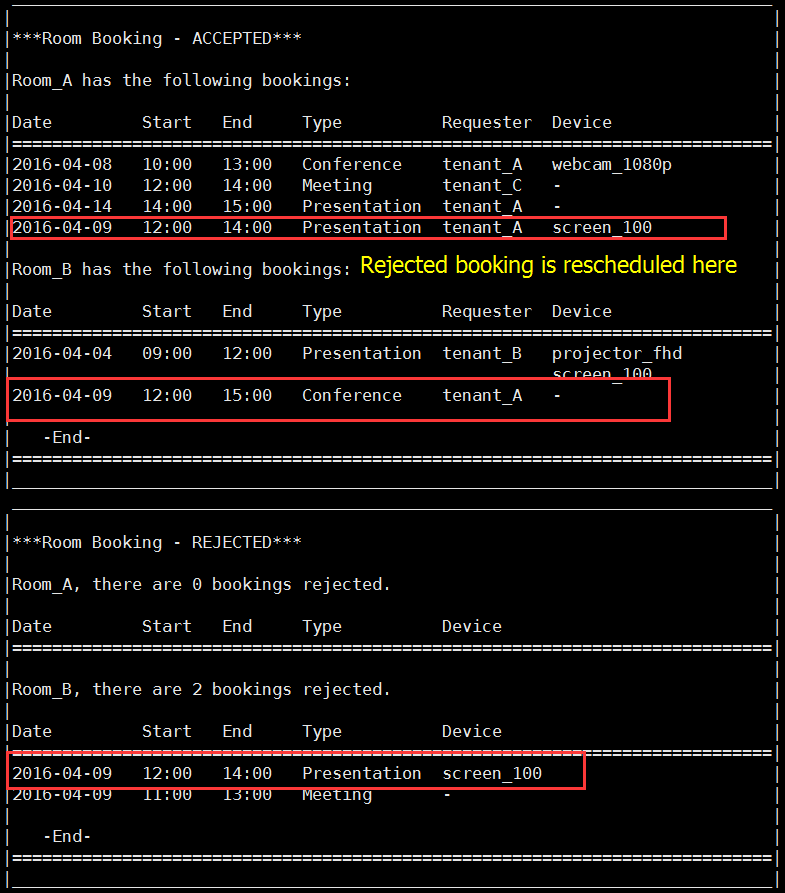
**Figure 3**

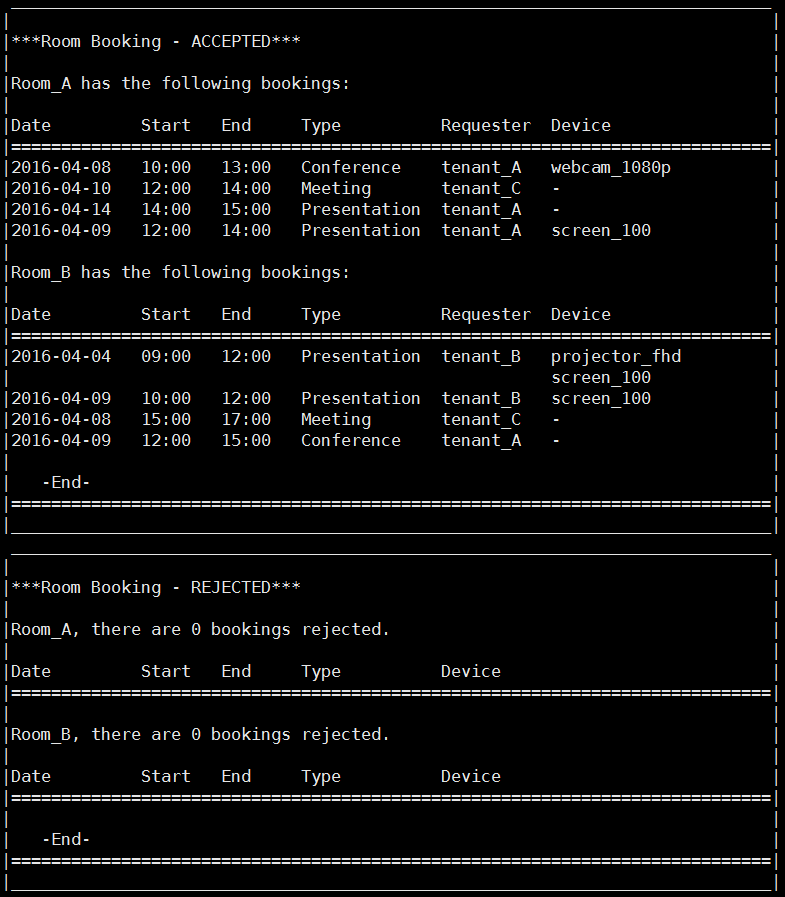


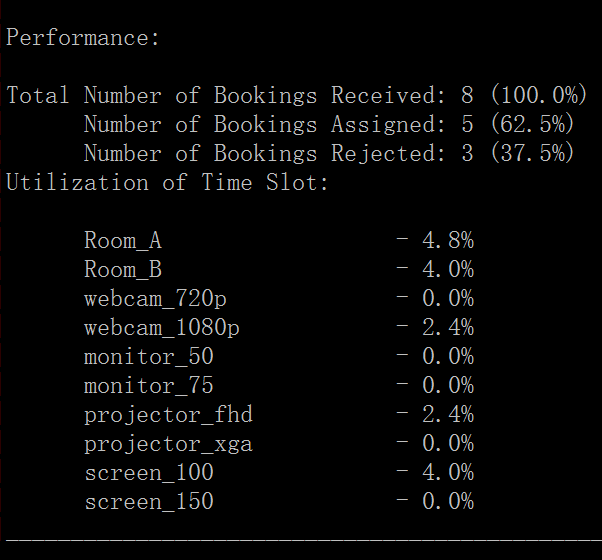
**Figure 4**

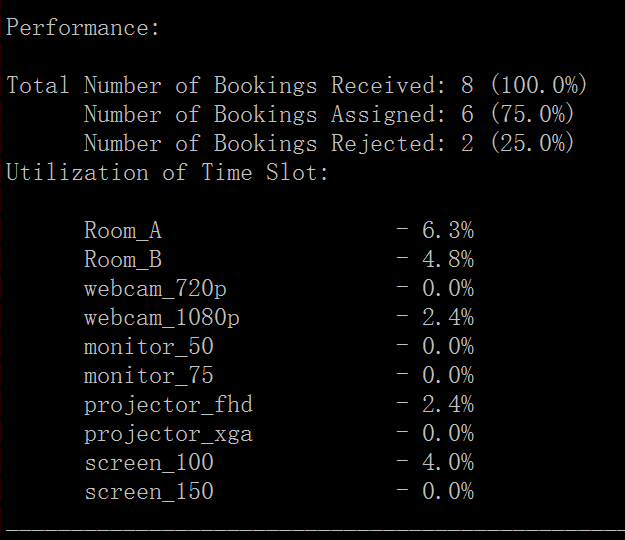
**Figure 5**

**Figure 6**

**Figure 7**

**Figure 8**

**Figure 9** 

**Figure 10** 

**Figure 11** 