模拟铁路版本 B

说明文档

2013-3-15

梁友
101250077

目录

1.设计说明

1.1 通信协议

结构体定义	**结构体说明**
typedef struct user_str{ char nickname[20]; char password[20]; }user;	用户帐号密码的结构体。
typedef struct station_str{ int train_station_id; char station_name[20]; char arrive_time[20]; int arrive_price; }station;	某个车次上的站点，包括ID、站点名称、到达该站点时间、从出发站达到该站点的累计票价。
typedef struct ticket_info_str{ char start_station_name[20]; char destination_station_name[20]; char departure_time[20]; }ticket_info;	用户需要查询的信息，包含出发站名称、到达站名称、出发日期。
typedef struct buy_ticket_info_str{ int train_number; char user_name[20]; char start_station_name[20];	用户买票所需要的信息，包含车次、用户名、出发站名称、到达站名称、出发日期。

char destination_station_name[20]; char departure_time[20]; }buy_ticket_info;	
typedef struct remain_ticket_str { int train_number; char start_time[20]; char arrive_time[20]; int price; int tickets; }remain_ticket;	服务器返回给客户端的车票信息，包含车次、出发时间、到达时间、票价、剩余车票数量。
typedef struct sold_ticket_str { char user_name[20]; char departure_time[20]; int train_number; char start_station_name[20]; char destination_station_name[20]; int sell_seat_number; }sold_ticket;	售票日志中的一条，包含用户名、出发日期、车次、出发站名称、到达站名称、所购车票座位号。

1.2 通信方式

系统采用 C/S 架构，本版本是网络版，进程间通信采用 socket 通信方式；

为支持多客户端，使用了 select 系统调用。

2.使用说明

2.1 服务器

包含文件：server.c　server.h　message.h

server.h： 服务器端头文件

message.h： 服务器和客户端通信结构体定义

server.c： 服务器主文件

编译方式：gcc –c server server.c

运行方式：./server

2.2 客户端

包含文件：client.c　client.h　message.h

client.h： 客户端头文件

message.h： 客户端和服务器通信结构体定义

client.c： 客户端主文件

编译方式：gcc –c client client.c

运行方式：./client

2.3 预存储说明

预先存在的一个账户用户名为“manager”,密码为”123”；

该账户为管理员账户，管理员账户不可以注册。

预先不存在任何站点、车次，必须由管理员登陆后录入。

预先不存在任何普通用户账户，必须注册。

3.实例界面

3.1 编译运行

```
ricky@ricky-OptiPlex-790:~$ gcc -o server '/home/ricky/桌面/my_code/server.c'
ricky@ricky-OptiPlex-790:~$ gcc -o client '/home/ricky/桌面/my_code/client.c'
```

```
ricky@ricky-OptiPlex-790:~$ ./server
```

```
ricky@ricky-OptiPlex-790:~$ ./client
```

3.2 管理员登陆

```
1 - login
2 - registe
3 - exit
choose one item:
1
nickname :
manager
password :
123
Welcome, manager!
```

3.3 添加站点

```
1 - add station
2 - add train
3 - first page
4 - exit
choose one item:
1
enter station name : (enter q to exit)
beijing
succeed!
enter station name : (enter q to exit)
shanghai
succeed!
enter station name : (enter q to exit)
nanjing
succeed!
enter station name : (enter q to exit)
q
```

3.4 添加车次

```
1 - add station
2 - add train
3 - first page
4 - exit
choose one item:
2
enter train number (between 1000 and 9999):
1000
ke hu duansucceed
enter station name along the train number (enter q to exit):
beijing
enter arrive time (like 18:32):
12:00
enter arrive price :
0
enter station name along the train number (enter q to exit):
shanghai
enter arrive time (like 18:32):
16:30
enter arrive price :
280
enter station name along the train number (enter q to exit):
nanjing
enter arrive time (like 18:32):
18:00
enter arrive price :
390
enter station name along the train number (enter q to exit):
q
```

3.5 用户注册

```
1 - login
2 - registe
3 - exit
choose one item:
2
nickname :
liang
password :
123
succeed!
```

3.6 购买车票

```
1 - buy tickets
2 - already bought
3 - first page
4 - exit
choose one item:
1
enter your start :
beijing
enter your destination :
shanghai
enter your departure date (like 20130226, must be 8-bit interger):
20130318
车次 出发时间 到达时间 票价 余票
1001  13:00    18:00   480   16
1000  12:00    16:30   280   16
enter the train_number you choose(enter 0 to quit):
1001
succeed!
```

3.7 查看已购车票

```
1 - buy tickets
2 - already bought
3 - first page
4 - exit
choose one item:
2
 日期     车次   出发站   达到站   座位号
20130318 1001 beijing shanghai   15
20130318 1000 beijing nanjing   15
```

4.文件存储说明

4.1 identity

保存普通用户的用户名和密码信息，其中用户名不可重复

identity

```
1 liang 123
2 you 123
```

4.2 station

保存站点信息，站点 ID 为介于 1000 和 9999 之间的一个不重复的四位整数

station

```
1 1000 beijing
2 1001 shanghai
3 1002 nanjing
4 1003 wuhan
```

4.3 train

保存车次信息，车次在每个站点有一个 ID，这个 ID 是一个八位整数；

前四位是车次 ID，后四位是站点 ID；

第二项是站点名称；

第三项是该车次到达该站点的时间；

第四项是车次从出发站到达该站的累计票价。

train

```
1 10001000 beijing 12:00 0
2 10001001 shanghai 16:30 280
3 10001002 nanjing 18:00 390
4 10011000 beijing 13:00 0
5 10011003 wuhan 16:50 350
6 10011001 shanghai 18:00 480
```

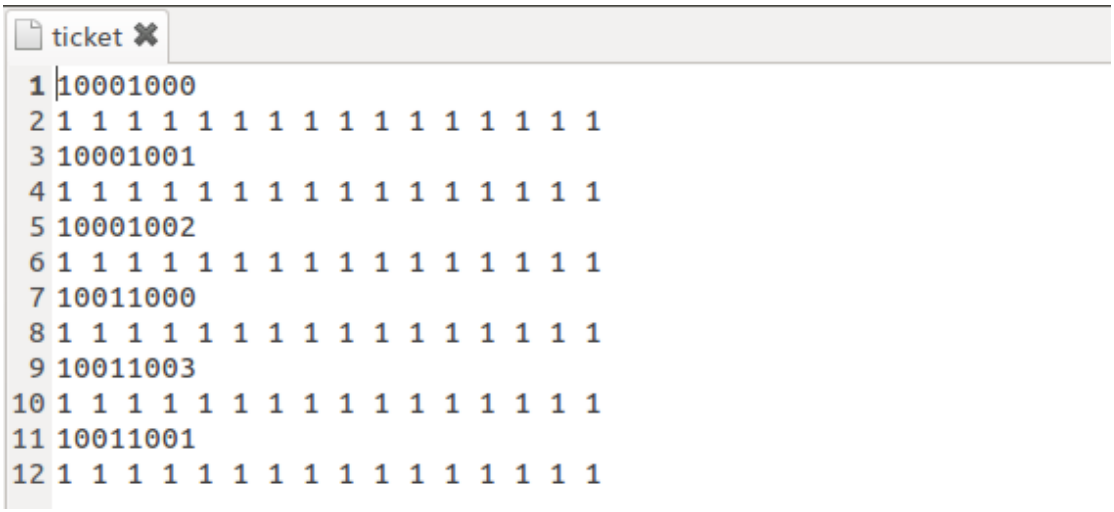
4.4 ticket

保存车票副本，每辆车座位设定为 16 个；

每两个 ID 之间有 16 个 0/1，表示从上一个 ID 到下一个 ID 座位；

如果是 1，表示该位置没人坐；如果是 0，表示该位置有人坐；

10001000 和 10001001 之间 16 个 1 表示，车次 1000 从站点 1000 到 1001 有 16 个座位。



4.5 20130318

按照日期名保存当日车票信息，标记规则同上；

10001000 和 10001001 之间 15 个 1 和一个 0 表示，车次 1000 从站点 1000 到 1001 有 16 个座位，其中第 16 个座位已经售出。

20130318

```
10001000
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
10001001
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
10001002
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
10011000
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
10011003
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
10011001
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

4.6 sold_tickets

按照日期名保存当日车票信息，标记规则同上；

保存售票记录，每个条目包含用户名、日期、车次、出发地、目的地、座位号；

第一行意为：liang 购买了 2013.03.15 车次 1000 从 beijing 到 shanghai 的票，

座位号为 15.

sold_tickets

```
liang 20130315 1000 beijing nanjing 15
you 20130318 1001 beijing shanghai 15
you 20130318 1000 beijing nanjing 15
```

5. 函数说明

5.1 辅助模块

函数或结构体定义	**函数或结构体说明**
typedef struct node_str{ int train_number; char start_time[8]; char arrive_time[8]; int price; int tickets; struct node_str *next; }node;	描述搜索满足条件的车次结果的结构体
typedef struct node_seat_str{ int seat_number; struct node_seat_str *next; }node_seat;	可以坐的位置的结构体
void copy_file(char *des, char *src);	将 src 中的内容复制到 des
node* insert_train_item(node *item_list, int new_train_number, char *new_start_time, char *new_arrive_time,	链表中插入一个新的车次及相关信息

int new_price, int new_tickets);	
node_seat* insert_seat_item(node_seat *item_list, int new_seat_number);	链表中插入一个新的座位
node_seat* delete_seat_item(node_seat *item_list, int old_seat_number);	链表中删除一个不符合要求的座位
void print_node(node *head);	打印满足搜索结果的车次的信息
void print_node_seat(node_seat *head);	打印满足条件的座位的信息

5.2 登录模块

函数或结构体定义	**函数或结构体说明**
void first_page();	主页，用户可选择注册登录，预先存在的是唯一一个管理员账户，用户名为 manager，密码为 123
void login();	登录判定
void registe();	注册判定
int search_identity(char *nickname, char *password);	搜索某个账户是否已经存在，存在返回 1

5.3 用户模块

函数或结构体定义	**函数或结构体说明**
void visit_func();	非管理员账户主功能界面
void buy_tickets();	买票
void sell_seat(int train_number, int start_station_id, int destination_station_id, char *departure_time);	自动选择座位，并把已被购买的座位标注信息
void view_tickets();	根据出发到达站，日期查看座位信息
node* find_useful_train(char *start_station_name, char *destination_station_name, char *departure_time);	查找满足用户搜索条件的所有车次及相关信息
node_seat* get_remaining_tickets(int train_number, int start_station_id, int destination_station_id, char *departure_time);	查看满足条件的余票信息，返回余票链表

5.4 管理员模块

函数或结构体定义	**函数或结构体说明**
void manager_func();	管理员主功能界面
void add_station();	添加站点，需要输入站点名称
int view_station(int visible);	查看站点信息的函数；如果参数为 1，打印所有站点信息，否则不打印； 返回最后一个站点的 ID，若没有站点返回 0
int search_station(char *station_name);	查看某站点是否存在，不存在文件返回 0，存在文件没有站点返回 1， 存在站点返回站点 ID
void add_train();	添加车次
int search_train(int train_number);	判断某个车次是否已经存在
void add_station_along_train(inttrain_number);	给某个车次添加所经过的站点