**实验报告：音乐厅门票订售系统**

专业：智能科学与技术 年级：2014级 姓名：冯立刚 学号： 22920142203809

1. **实验要求**

1、利用IPC机制仿真一个音乐厅门票订售系统；该系统包含若干售票代

理，这些代理连接到一个票务中心计算机。

2、每个代理可以看成是一个可并发执行的进程，如果某个代理企图订/

售一个已被占用的座位，系统应该禁止该操作并打印“座位已占用”的提示信息。如果某交易涉及对不存在的座位进行操作，则系统也应该打印出错信息。

3、需要注意的是：购票之前无需先预订，而取消操作只有到涉及的座位被预订但尚未售出时候才有效。

4、为使结果显示更明晰，系统在每个交易完成后都应该给出交易的信息摘要，而在系统结束后则应该显示出座位的订售情况（包含客户信息）。

# 二、实验环境

1、系统平台：Ubuntu 64位,Windows10 64位;

2、集成开发环境：Visual Studio Community 2017;

# 三、实验内容

1、程序框架：

**票务中心**

**（Ticket\_Center）**

**售票代理4**

**（Agent）**

**售票代理3**

**（Agent）**

**售票代理2**

**（Agent）**

**售票代理1**

**（Agent）**

**a.**音乐厅座位状态用一个二维整型数组seat存储（1：可用，-1：预定0:

售出）,共享节中的数据变量**seat**（座位状态），**flag**，**No**（代理商编号），**Lock**(互斥访问标记)等会被多个进程共享，其值若在一个进程中被改变，则该操作对其它进程可见；

*//--------------共享节------------------------------*

*#pragma data\_seg("Shared")*

*volatile int seat[N][M] = {0};//存放音乐厅座位状态*

*volatile int flag = 1;//初始化标记*

*volatile int No = 1; //售票终端编号*

*volatile int Lock = 0;//0：文件关闭，1:文件打开*

*#pragma data\_seg*

*#pragma comment(linker,"/Section:Shared,RWS")*

**b.**票务中心包含以下操作：*Set\_ticket():购买票；Set\_reserve():预定票；Set\_cancel():取消票；Show\_ticket():显示用户购买的票。Show\_reserve():显示用户预定的票*；*Get\_state():获得音乐厅座位的状态。*

票务中心涉及到对座位状态seat、顾客预定单reserve、顾客购买单ticket的操作，是执行售票任务的最终操作者。

基本操作为通过seat[][]=value改变座位状态，然后从文件中载入顾客（预定/购买）订单，并用新的用户操作改变订单状态，然后将订单写入文件，并返回操作结果。至此通过共享节、文件、锁机制实现了进程之间的数据共享。

*seat[alphabet\_To\_number(in)][im] = 0;*

*getticket();*

*if (Iterator\_ticket != ticket\_record.end())//在订单中找到了顾客姓名*

*ticket\_record[customer].push\_back(temp);*

*else //没有在订单中找到顾客姓名*

*ticket\_vector.push\_back(temp);*

*ticket\_record.insert(map<string,vector<string>>::value\_type(customer, ticket\_vector));*

*}*

*giveticket();*

**c.**售票代理Agent有一个成员变量Ticket\_center,借此可以访问票务中心提供的接口函数，进行完整的音乐厅票务代理操作。

*class Agent*

*{*

*private:*

*Ticket\_center T;*

*public:*

*Agent(Ticket\_center TT)*

*{*

*T = TT;*

*}*

*int reserve(int rows, int columns, string customer)*

*{*

*return T.set\_reserve(rows, columns, customer);*

*}*

*int ticket(char rows, int columns, string customer)*

*int cancel(int rows, int columns, string customer)*

*void show\_ticket(string customer)*

*void show\_reserve(string customer)*

*int get\_state(char rows, int columns)*

*}*

2、程序运行流程

程序启动

多个售票代理 Agent 1, 2, 3, 4, 5 ···, k 等待用户输入

用户输入

取消cancel

显示

show

返回操作结果给用户

购票ticket

订票

reserve

参数传递给Agent

访问票务中心Ticket\_center

票务中心更新 seat[][] reserve & ticket 订单

# 四、项目测试

编写测试脚本启动K个售票代理终端，在脚本中我们企图启动十几个终端进程，但是由于程序限定最多只能有K个售票代理，所以该脚本最多启动K个进程。

*@echo off*

*start musichall.exe*

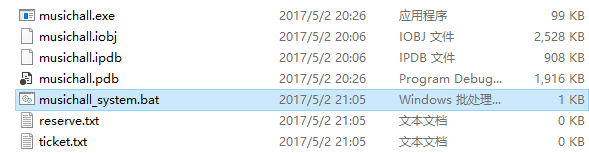
*ping 127.1 -n 2 >nul*

*···*

测试项目包括购票、订票、取消订票、显示顾客订单的测试，以及多个售票代理终端数据共享同步的测试。

1. 程序正常启动，开启多个售票代理终端：

a.程序包目录如下：



b.执行测试脚本启动多个终端：



可以清晰的看到每个终端的提示信息中显示了售票代理号，共打开6个终端，这也证明了进程间实现了数据共享。

## 2、购票测试（N=26，M=10,K=6）

测试用例：

ticket 4 gatsby；

ticket G gatsby;

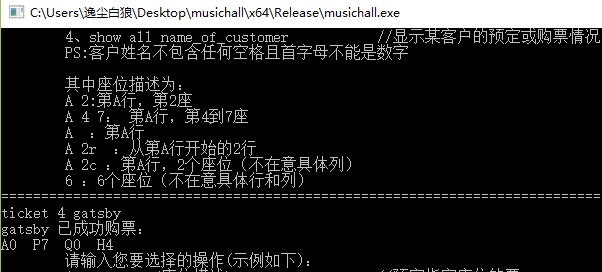
ticket B 4 gatsby;

ticket C 4c gatsby;

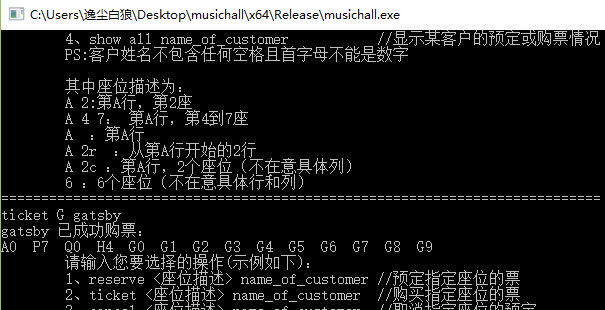
ticket D 4r gatsby;

ticket F 4 8 gatsby;

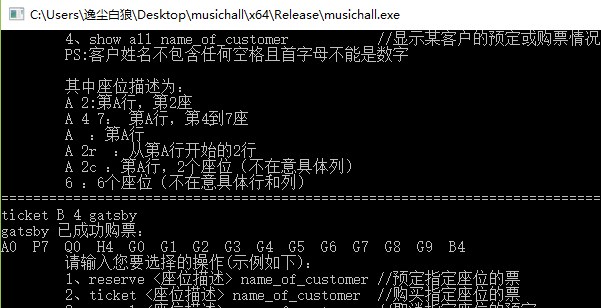
ticket 4 gatsby：



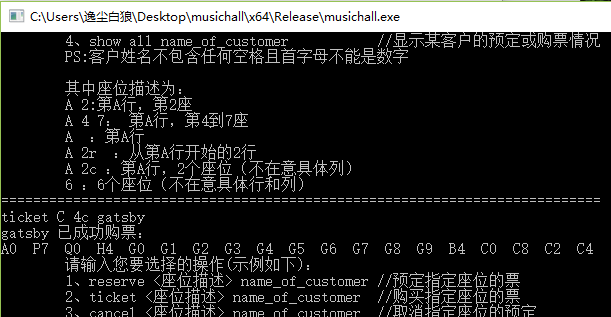
ticket G gatsby：



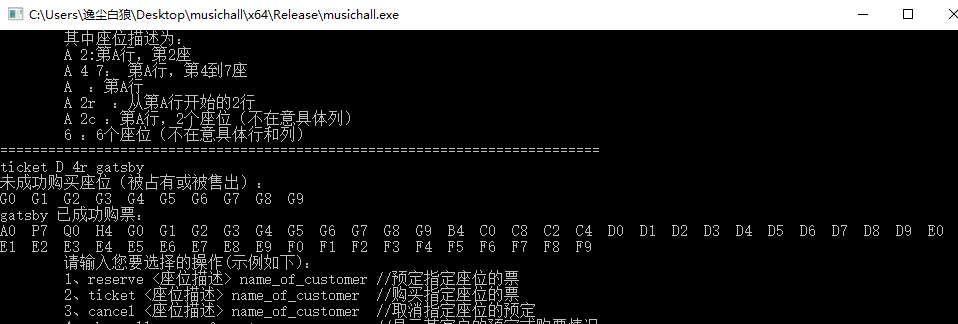
tikket B 4 gatsby：



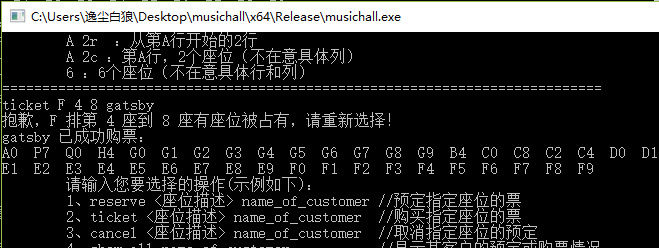
ticket C 4c gatsby：



ticket D 4r gatsby：



ticket F 4 8 gatsby：



## 3、预定票和取消票测试

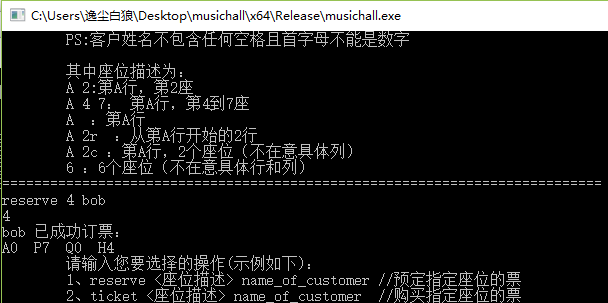
测试用例：

reserve 4 bob;

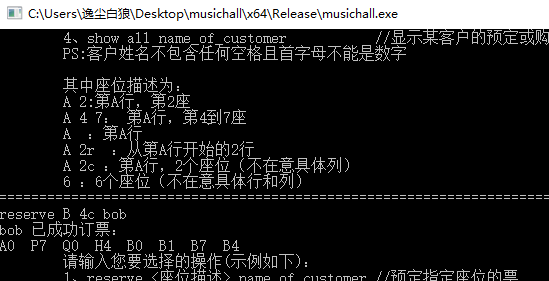
reserve B 4c bob;

cancel \*\* bob;

reserve 4 bob:



reserve B 4c bob:



cancel \*\* bob:



## 4、显示顾客订单和代理数据共享同步测试：

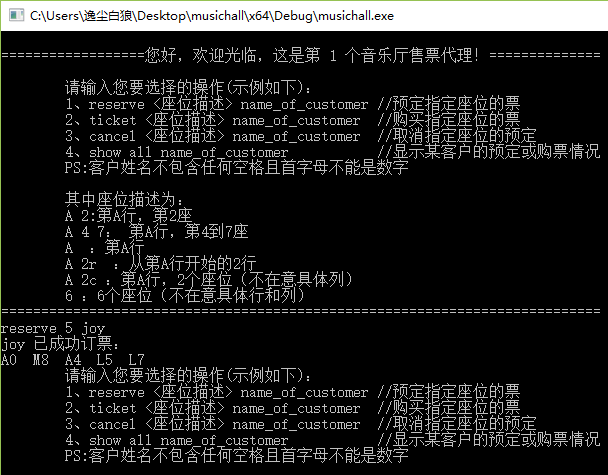
测试用例：

reserve 5 joy;

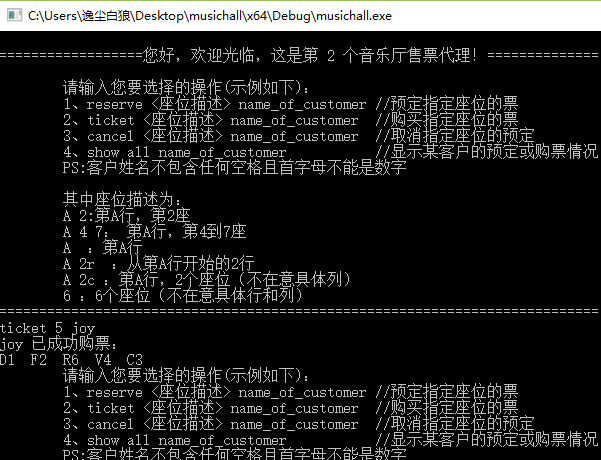
ticket 5 joy;

show all joy;

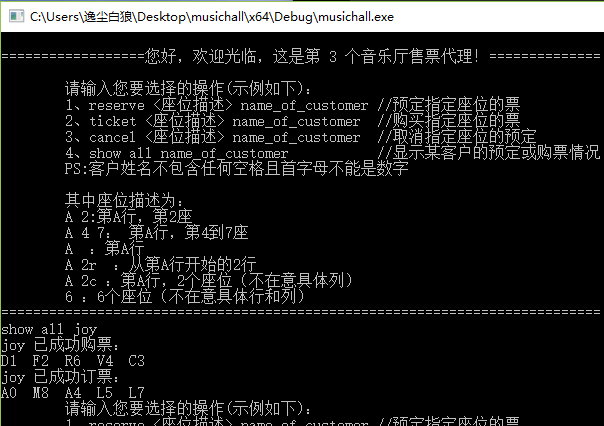
1号代理 reserve 5 joy:



2号代理 ticket 5 joy:



3号代理show all joy:



# 五、实验心得

之前我写的许多程序基本上都是单进程的，几个函数要共享变量就使用全局变量或其他方法，相对来说比较简单和容易理解。第一次写多进程程序，并且还要实现多个进程之间的数据共享，想想还是比较麻烦的。然而通过百度和其他工具我找到了实现共享的方法，有许多种，最终选择了最好实现的一种方法完成了多进程程序的编写。

对程序的编写也有了更近一步的理解，编写程序之前对项目需求的透彻理解可以辅助建立完整的框架，并对选择什么工具和方法提供更好的参考，这一点很重要。