附录

1. 测试视频

测试视频是对整个航班购票模拟系统的测试，包括服务端、售票端和购票端的测试过程。视频已上传至哔哩哔哩弹幕网，请点击链接查看。

视频链接：https://www.bilibili.com/video/BV14T4y1J7bt/

2. 实验代码

实验代码已经上传到了 Gitee 仓库，包含了服务端、售票端和购票端的 QT 工程代码以及 Linux C 底层代码。

代码仓库链接：https://gitee.com/zhj0125/TicketingSystem

由于我们是直接在课本例程基础上进行的代码移植，直接将嵌入式代码导入到了 QT 中，因此下面只列出了 QT 程序中的所有代码。

2.1 TicketingSystem_Server 服务端代码

（1）TicketingSystem_Server.pro 文件

```
QT       += core gui
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
CONFIG += c++11

# The following define makes your compiler emit warnings if you use
# any Qt feature that has been marked deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS
# You can also make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000
# disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    globalapi.h \
    mainwindow.h \
    servicethread.h \
```

```
    threadbuff.h \
    ticket.h

FORMS += \
    mainwindow.ui
# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

RESOURCES += \
    Icon.qrc

LIBS += \
    -lmysqlclient
```

（2）globalapi.h 文件

```
#ifndef    __GLOBALAPI_H
#define    __GLOBALAPI_H
#define GTK_ENABLE_BROKEN
#include <sys/types.h>          /*基本的系统数据类型*/
#include <sys/socket.h>         /*基本的套接字的定义*/
#include <sys/time.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/un.h>
#include <string.h>
#include <pthread.h>
#include <mysql/mysql.h>

/*服务器端使用的端口*/
#define    SERVER_PORT_NO          8889

/*客户端与服务器端使用的消息类型定义*/
#define    INITIAL_VALUE           65535

/*客户端使用的消息代码含义*/
#define    DISCONNECT              0
#define    BUY_TICKET              1
#define    INQUIRE_ONE             2
```

```
#define   INQUIRE_ALL            3
#define   ADD_TICKET             4
#define   UPDATE_TICKET          5
#define   DELETE_TICKET          6

/*服务器端使用的消息代码含义*/
#define   BUY_SUCCEED            255
#define   BUY_FAILED             256
#define   INQUIRE_SUCCEED        257
#define   UNKNOWN_CODE           258
/* 服务器与客户端使用的消息结构定义 */
struct stMessage {
   // 消息类型。客户端可以取值为 DISCONNECT：断开连接
   // BUY_TICKET：购买机票
   // INQUIRE_ONE：查询特定航班机票
   // INQUIRE_ALL：查询所有航班机票
   unsigned int    msg_type;
   unsigned int    flight_ID;             // 航班号
   unsigned int    ticket_num;            // 机票张数
   unsigned int    ticket_total_price;    // 机票价钱
} message;

void init_message(){
   message.msg_type=INITIAL_VALUE;
   message.flight_ID=0;
   message.ticket_num=0;
   message.ticket_total_price=0;
}

/* 服务器端的线程缓冲区的最大数量 */
#define   THREAD_BUFF_NUM 128

/* 提示信息输出 */
#define   INFO_NUM               10
#define   INFO_OCCUPIED          1
#define   INFO_FREED             0

struct info_t {
   int status;            /* INFO_OCCUPIED or INFO_FREED */
   char msg[512];         /* contents of message */
}  info[INFO_NUM];
pthread_mutex_t  info_mutex;

/* 初始化界面输出信息缓冲区 */
void init_info(){
```

```
    int i;
    for(i=INFO_NUM;i>0;i--){
       info[i-1].status=INFO_FREED;
    }
    sprintf(info[i-1].msg," ");
}

/* 分配一个空闲的界面输出信息缓冲区，如果没有空闲的缓冲区则返回-1 */
int get_free_info(){
    int i,ret;
    /* 注意对互斥锁的操作，这些操作必须是成对的（加锁和解锁），否则会发生死锁的情况 */
    pthread_mutex_lock(&info_mutex);
    for(i=0;i<INFO_NUM; i++){
            if(info[i].status==INFO_FREED) {
                    ret=i;
                    pthread_mutex_unlock(&info_mutex);
                    break;
            }
    }
    if(i==INFO_NUM) {
            ret=-1;
            pthread_mutex_unlock(&info_mutex);
    }
    return ret;
}

/* 释放界面输出信息缓冲区，对 info_status 的访问同样需要使用互斥保护 */
void free_info(int index){
    pthread_mutex_lock(&info_mutex);
    if(info[index].status==INFO_OCCUPIED){
            info[index].status=INFO_FREED;
    }
    pthread_mutex_unlock(&info_mutex);
}

void add_info(char *src){
    int i;
    while((i=get_free_info())==-1){
            sleep(1);
    }
    // 添加消息
    pthread_mutex_lock(&info_mutex);
    info[i].status=INFO_OCCUPIED;
    strcpy(info[i].msg, src);
    pthread_mutex_unlock(&info_mutex);
```

```
}
#endif
```

（3）mainwindow.h 文件

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QDialog>
#include <QFormLayout>
#include <QList>
#include <QLineEdit>
#include <QDialogButtonBox>
#include <QTimer>
#include <QDebug>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    void display_info(QString msg);
    void enable_button(bool boolean);

private slots:
    void on_action_start_triggered();
    void on_action_stop_triggered();
    void on_action_exit_triggered();
    void on_action_inquireone_triggered();
    void on_action_inquireall_triggered();
    void on_action_show_triggered();
    void on_action_about_triggered();

private:
    Ui::MainWindow *ui;

protected:
    void paintEvent(QPaintEvent *event); //添加重绘事件
};
#endif // MAINWINDOW_H
```

（4）servicethread.h 文件

```
/*servicethread.h*/
#ifndef    __SERVICE_THREAD_H
#define    __SERVICE_THREAD_H
#include "ticket.h"
#include "threadbuff.h"
#include "mainwindow.h"

/* thread_err: 服务线程的错误处理函数，由于服务器端使用的是多线程技术，服务线程发生错误时，不能像在多进程的情况下，简单地调用 exit()终止进程 */
/* 在多线程下，服务线程必须将使用的资源释放后，调用 pthread_exit()退出，并且在需要进行线程间同步的情况下，还需要做一些线程同步的工作，才能退出 */
/* 这个特点在多线程编程中是非常重要的 */
static void thread_err(char *s, int index){
   char msg[512];
   /*获取空闲的界面输出信息缓冲区，如果没有空闲的,延迟一段时间后继续获取*/
   sprintf(msg,"线程 %d 发生致命错误：,%s\n",(unsigned short)pthread_self(),s);
   add_info(msg);
   //info_print(strmsg,serverwindow);
   /*释放线程使用的线程缓冲区*/
   free_buff(index);
   pthread_exit(NULL);
}

/*service_thread:服务线程的线程函数。服务线程根据函数的参数中获取自身使用的线程缓冲区的序号，而后根据这个序号从线程缓冲区中获取需要的参数*/
void * service_thread(void *p){
   int                  conn_fd;
   int                  buff_index;
   char                 send_buf[1024],recv_buf[1024];
   int                  ret,i;
   struct sockaddr_in   peer_name;
   socklen_t            peer_name_len;
   unsigned int         required_ticket_num;
   int                  pos;
   thread_buff_struct *pstruct;
   char msg[512];

   /*获取线程使用的线程缓冲区的序号*/
   pstruct=(thread_buff_struct *)p;
   buff_index=pstruct->buff_index;
   pstruct->tid=pthread_self();
```

```
        /*从线程缓冲区中获取通信使用的套接字描述符*/
        conn_fd=pstruct->conn_fd;

        /*打印远端主机地址*/
        peer_name_len=sizeof(peer_name);
        ret=getpeername(conn_fd,(struct sockaddr*)&peer_name, &peer_name_len);
        if(ret==-1){
           thread_err((char*)"获取远端主机地址出错",buff_index);
        }

        sprintf(msg,"新连接-->线程 ID：%d, 连接 ID：%d, 线程缓冲区索引号：%d,
远端地址：%s, 端口号：%d\n",(unsigned short)pstruct->tid,conn_fd, buff_index,\
      inet_ntoa(peer_name.sin_addr), ntohs(peer_name.sin_port));
        add_info(msg);
        while(1) {
           /* 从网络中获取数据记录 */
           ret=recv(conn_fd,recv_buf,sizeof(message),0);
           /* 接收出错 */
           if(ret==-1) {
                sprintf(msg,"线程：%d 在连接：%d 接收出错。连接将关闭。\n",\
                (unsigned short)pstruct->tid, conn_fd);
                add_info(msg);
                thread_err(msg, buff_index);
           }
           /* ret==0 说明客户端连接已关闭 */
           if(ret==0) {
                sprintf(msg,"线程 %d 的连接( ID：%d ) 客户端已关闭。服务器端连
接也将关闭。\n",(unsigned short)pstruct->tid, conn_fd);
                add_info(msg);
                close(conn_fd);
                free_buff(buff_index);
                pthread_exit(NULL);
           }

           /* ret 为其他值说明接收到了客户端的请求消息 */
           init_message();
           memcpy(&message,recv_buf,sizeof(message));
           MYSQL mysql;
           MYSQL_RES * result;
           char sqlstr[100];
           switch(message.msg_type) {
              case DISCONNECT:
                sprintf(msg,"线程 %d 的连接(ID： %d ) 客户端已关闭。服务器端连接
也将关闭。\n",(unsigned short)pstruct->tid, conn_fd);
```

```
                add_info(msg);
                close(conn_fd);
                free_buff(buff_index);
                pthread_exit(NULL);
                break;

            case BUY_TICKET :
                read_ticket_list();    // 读取数据库机票信息
                mysql_init(&mysql);
                mysql_real_connect(&mysql, "localhost", "zhj", "666588", "linux", 0,
NULL, 0);
                mysql_query(&mysql, "select * from tickets");
                result = mysql_store_result(&mysql);//将查询的全部结果读取到客户端
                numRows = mysql_num_rows(result);  //统计结果集的行数
                for(i=0; i<numRows; i++) {
                        pthread_mutex_lock(&ticket_list[i].ticket_mutex);
                        if(ticket_list[i].flight_ID==message.flight_ID) {
                           //剩余票数大于请求票数
                           if(ticket_list[i].ticket_num>=message.ticket_num) {
                              message.msg_type=BUY_SUCCEED;
                              message.ticket_total_price=\
                              message.ticket_num*ticket_list[i].ticket_price;
                              ticket_list[i].ticket_num-=message.ticket_num;
                              /* 更新数据库机票信息 */
                              sprintf(sqlstr, "update tickets set ticket_num = %d where
flight_ID = %d",ticket_list[i].ticket_num,i+1);
                              mysql_query(&mysql,sqlstr);    // 执行更新语句
                              mysql_free_result(result);
                              mysql_close(&mysql);
                              pthread_mutex_unlock(&ticket_list[i].ticket_mutex);
                              sprintf(msg,"客户端 %s 购买机票成功！航班号：%d, 票
数：%d, 总票价：%d\n",inet_ntoa(peer_name.sin_addr),message.flight_ID,\
                                message.ticket_num, message.ticket_total_price);
                                add_info(msg);
                             memcpy(send_buf,&message,sizeof(message));
                             ret=send(conn_fd, send_buf, sizeof(message), 0);
                             if(ret<0){
                                thread_err((char*)"发送数据出错\n", buff_index);
                             }
                             break;
                           } else {          //剩余票数不足，购买失败
                             message.msg_type=BUY_FAILED;
                             required_ticket_num=message.ticket_num;
                             message.ticket_num=ticket_list[i].ticket_num;
                             pthread_mutex_unlock(&ticket_list[i].ticket_mutex);
```

```
                    sprintf(msg,"客户端 %s 购买机票失败！航班号：%d, 剩余票数：%d, 请求票数：%d\n",inet_ntoa(peer_name.sin_addr),message.flight_ID, message.ticket_num,required_ticket_num);
                    add_info(msg);
                    memcpy(send_buf,&message,sizeof(message));
                    ret=send(conn_fd, send_buf, sizeof(message), 0);
                    if(ret<0){
                       thread_err((char*)"发送数据出错\n", buff_index);
                    }
                    break;
                 }
              }
              pthread_mutex_unlock(&ticket_list[i].ticket_mutex);
           }
           break;

        case INQUIRE_ONE:
           read_ticket_list();    // 读取数据库机票信息
           update_ticket_number();
           for(i=0; i<numRows; i++) {
              pthread_mutex_lock(&ticket_list[i].ticket_mutex);
              if(ticket_list[i].flight_ID==message.flight_ID) {
                 message.msg_type=INQUIRE_SUCCEED;
                 message.ticket_num=ticket_list[i].ticket_num;
                 message.ticket_total_price=ticket_list[i].ticket_price;
                 pthread_mutex_unlock(&ticket_list[i].ticket_mutex);
                 sprintf(msg,"客户端 %s 查询航班号：%d 成功！\n",\
                  inet_ntoa(peer_name.sin_addr),message.flight_ID);
                 add_info(msg);
                 memcpy(send_buf,&message,sizeof(message));
                 ret=send(conn_fd, send_buf, sizeof(message), 0);
                 if(ret<0)
                    thread_err((char*)"发送数据出错\n", buff_index);
                 break;
              }
                    pthread_mutex_unlock(&ticket_list[i].ticket_mutex);
           }
           break;

        case INQUIRE_ALL:
                        pos=0;
           read_ticket_list();    // 读取数据库机票信息
           update_ticket_number();
           for(i=0; i<numRows; i++) {
              pthread_mutex_lock(&ticket_list[i].ticket_mutex);
              message.msg_type=INQUIRE_SUCCEED;
```

```
                message.flight_ID=ticket_list[i].flight_ID;
                message.ticket_num=ticket_list[i].ticket_num;
                message.ticket_total_price=ticket_list[i].ticket_price;
                pthread_mutex_unlock(&ticket_list[i].ticket_mutex);
                memcpy(send_buf+pos,&message,sizeof(message));
                pos+=sizeof(message);
            }
            sprintf(msg,"客 户 端 %s 查 询 所 有 航 班 号 成 功 ！
\n",inet_ntoa(peer_name.sin_addr));
            add_info(msg);
            ret=send(conn_fd, send_buf, pos, 0);
            if(ret<0){
                thread_err((char*)"发送数据出错\n", buff_index);
            }
            break;

        case ADD_TICKET:
            read_ticket_list();    // 读取数据库机票信息
            mysql_init(&mysql);
            mysql_real_connect(&mysql, "localhost", "zhj", "666588", "linux", 0,
NULL, 0);
            sprintf(sqlstr, "insert into tickets values(%d, %d, %d)", message.flight_ID,
message.ticket_num, message.ticket_total_price);
            mysql_query(&mysql,sqlstr);    // 执行更新语句
            mysql_close(&mysql);
            break;

        case UPDATE_TICKET:
            read_ticket_list();    // 读取数据库机票信息
            mysql_init(&mysql);
            mysql_real_connect(&mysql, "localhost", "zhj", "666588", "linux", 0,
NULL, 0);
            sprintf(sqlstr, "update tickets set ticket_price = %d where flight_ID = %d",
message.ticket_total_price, message.flight_ID);
            mysql_query(&mysql,sqlstr);    // 执行更新语句
            sprintf(sqlstr, "update tickets set ticket_num = %d where flight_ID = %d",
message.ticket_num, message.flight_ID);
            mysql_query(&mysql,sqlstr);    // 执行更新语句
            mysql_close(&mysql);
            break;

        case DELETE_TICKET:
            read_ticket_list();    // 读取数据库机票信息
            mysql_init(&mysql);
            mysql_real_connect(&mysql, "localhost", "zhj", "666588", "linux", 0,
```

```
NULL, 0);
                sprintf(sqlstr, "delete from tickets where flight_ID = %d",
message.flight_ID);
                mysql_query(&mysql,sqlstr);    // 执行更新语句
                mysql_close(&mysql);
                break;

            default:
                message.msg_type=UNKNOWN_CODE;
                memcpy(send_buf, &message,sizeof(message));
                ret=send(conn_fd, send_buf, sizeof(message), 0);
                if(ret<0){
                   thread_err((char*)"发送数据出错\n", buff_index);
                }
          }
      }
   }
   #endif
```

（5）threadbuff.h 文件

```
/*threadbuff.h*/
#ifndef   __THREAD_BUFF_H
#define   __THREAD_BUFF_H
#include "globalapi.h"

/*定义线程缓冲区的使用状态*/
#define   BUFF_OCCUPIED   1
#define   BUFF_FREED      0

/*线程缓冲区结构*/
typedef struct thread_buff_struct_t {
   int buff_index;                  // 线程缓冲区的索引号
   int tid;                         // 保存对应线程的线程号
   unsigned long ip_addr;           // 保存对应的客户机的 IP 地址
   int      conn_fd;                // 该线程使用的连接套接字描述符
   int      buff_status;            // 线程缓冲区的状态
} thread_buff_struct;
thread_buff_struct thread_buff[THREAD_BUFF_NUM];

/* 用于线程缓冲区互斥使用的互斥锁 */
/* 由于当主线程分配线程缓冲区时需要检测 buff_status 变量的值，而服务线程
在退出前，需要将它使用的线程缓冲区释放，所谓释放就是需要修改 buff_status 变
量的值，所以主线程和服务线程间需要对 buff_status 进行互斥，可以为每一个
buff_status 变量设置一个互斥锁，但这样需要较多的系统资源。这里只使用了一个
```

```
互斥锁来对结构数组中的所有 buff_status 变量进行互斥保护。*/
    pthread_mutex_t buff_mutex;

    /* 初始化线程缓冲区 */
    void init_thread_buff(){
      int index;
      for(index=0; index<THREAD_BUFF_NUM;index++) {
            thread_buff[index].tid=-1;
            thread_buff[index].buff_status=BUFF_FREED;
      }
    }

    /* 分配一个空闲的线程缓冲区，如果没有空闲的缓冲区则返回-1 */
    int get_free_buff(){
      int i,ret;
      /*注意对互斥锁的操作，这些操作必须是成对的（加锁和解锁），否则会发生死锁的情况*/
      pthread_mutex_lock(&buff_mutex);
      for(i=0;i<THREAD_BUFF_NUM; i++)
            if(thread_buff[i].buff_status==BUFF_FREED) {
                  ret=i;
                  pthread_mutex_unlock(&buff_mutex);
                  break;
            }
      if(i==THREAD_BUFF_NUM) {
            ret=-1;
            pthread_mutex_unlock(&buff_mutex);
      }
      return ret;
    }

    /* 释放线程缓冲区，对 buff_status 的访问同样需要使用互斥保护 */
    void free_buff(int index){
      pthread_mutex_lock(&buff_mutex);
      if(thread_buff[index].buff_status==BUFF_OCCUPIED){
            thread_buff[index].buff_status=BUFF_FREED;
      }
      pthread_mutex_unlock(&buff_mutex);
    }

    /* 检查线程缓冲区中是否重复连接，可能客户端的通信进程终止后重新启动 */
    /* 此时应当终止原来它所对应的服务线程，再重新创建一个服务线程，并为这个新的服务线程分配线程缓冲区 */
    void check_connection(unsigned long ip_addr){
      int i;
```

```
    struct in_addr in;
    char msg[512];
    /* 检查所有的线程缓冲区 */
    pthread_mutex_lock(&buff_mutex);
    for(i=0;i<THREAD_BUFF_NUM;i++) {
        /* 发现重复连接 */
        if((thread_buff[i].buff_status!=BUFF_FREED)&&thread_buff[i].ip_addr\
            ==ip_addr) {
            in.s_addr=htonl(ip_addr);
            sprintf(msg,"重复连接：%s，旧连接将关闭！\n",inet_ntoa(in));
            add_info(msg);
            pthread_cancel(thread_buff[i].tid);
            pthread_join(thread_buff[i].tid,NULL);
        /* 退出的线程不释放它的缓冲区，释放工作由主线程来处理 */
            thread_buff[i].tid=0;
            thread_buff[i].buff_status=BUFF_FREED;
            close(thread_buff[i].conn_fd);
        }
    }
    pthread_mutex_unlock(&buff_mutex);
}

#endif
```

（6）ticket.h 文件

```
/*ticket.h*/
#ifndef    __TICKET_H
#define    __TICKET_H
#include "globalapi.h"
#define    FLIGHT_NUM          50       // 航班总数
int numRows = 0;

/* 机票的简单描述，flight_ID 表示航班号，ticket_num 表示机票剩余票数 */
typedef struct ticket_struct_t {
    unsigned int flight_ID;
    unsigned int ticket_num;
    unsigned int ticket_price;                // 票价
    /*多个线程操作时，必须对机票的剩余数量进行保护。由于这样的操作比较
频繁，所以应当对每一个 ticket_num 使用不同的互斥锁，否则对线程间并行性有较
大影响。*/
    pthread_mutex_t        ticket_mutex;
} ticket_struct;
ticket_struct ticket_list[FLIGHT_NUM];
```

```
/* read_ticket_list:初始化 ticket_list 数组 */
void read_ticket_list(){
   MYSQL mysql;
   MYSQL_RES * result;
   MYSQL_ROW row;
   int numRows = 0;
   int i = 0;
   mysql_init(&mysql);
   mysql_real_connect(&mysql, "localhost", "zhj", "666588", "linux", 0, NULL, 0);
   mysql_query(&mysql, "select * from tickets");       // 调用 mysql_store_result 之
前必须检索数据库
   result = mysql_store_result(&mysql);          // 将查询的全部结果读取到客户端
   numRows = mysql_num_rows(result);          // 统计结果集的行数
   if(result){
      for(i=0;i<numRows;i++){
         if((row = mysql_fetch_row(result)) != NULL){
            ticket_list[i].flight_ID = atoi(row[0]);
            ticket_list[i].ticket_num = atoi(row[1]);
            ticket_list[i].ticket_price = atoi(row[2]);
         }
      }
   }
   mysql_free_result(result);                    // 释放 result 空间，避免内存泄漏
   mysql_close(&mysql);
   return;
}

void update_ticket_number(void){
   MYSQL mysql;
   MYSQL_RES * result;
   mysql_init(&mysql);
   mysql_real_connect(&mysql, "localhost", "zhj", "666588", "linux", 0, NULL, 0);
   mysql_query(&mysql, "select * from tickets");
   result = mysql_store_result(&mysql);          // 将查询的全部结果读取到客户端
   numRows = mysql_num_rows(result);          // 统计结果集的行数
   mysql_free_result(result);
   mysql_close(&mysql);
}
#endif
```

（7）main.cpp 文件

```
#include "mainwindow.h"
#include <QApplication>
int main(int argc, char *argv[])
{
```

```
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

（8）mainwindow.cpp 文件

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "servicethread.h"

int listen_fd, conn_fd;                    // 监听 socket，连接 socket
struct sockaddr_in server, cli_addr;       // 服务器地址信息，客户端地址信息
int ret, buffer_index, i;
socklen_t cli_len;
unsigned long ip_addr;
int flag = 1;
pthread_t listentid,servicetid;            // 监听线程 ID,服务线程 ID；
int isserveropened = false;                // 服务器端是否开启标志位

void * listen_thread(void*)
{
    char msg[512];
    while(1) {
        /* 接受远端的 TCP 连接请求 */
        cli_len = sizeof(cli_addr);
        conn_fd = accept(listen_fd,(struct sockaddr *)&cli_addr, &cli_len);
        if(conn_fd < 0){
            continue;
        }
        ip_addr = ntohl(cli_addr.sin_addr.s_addr);
        /* 检测重复连接 */
        //check_connection(ip_addr);
        /* 分配线程缓冲区 */
        buffer_index = get_free_buff();
        if(buffer_index < 0) {
            sprintf(msg, "没用空闲的线程缓冲区。\n");
            add_info(msg);
            close(conn_fd);
            continue;
        }
        /* 填写服务线程需要的信息 */
        pthread_mutex_lock(&buff_mutex);
        thread_buff[buffer_index].buff_index=buffer_index;
        thread_buff[buffer_index].ip_addr=ip_addr;
```

```
            thread_buff[buffer_index].conn_fd=conn_fd;
            thread_buff[buffer_index].buff_status=BUFF_OCCUPIED;
            pthread_mutex_unlock(&buff_mutex);
            /* 创建新的服务线程 */
            ret=pthread_create(&servicetid,  NULL,  service_thread,
&thread_buff[buffer_index]);
            if(ret==-1) {
               sprintf(msg,"创建服务线程出错！\n");
               add_info(msg);
               close(conn_fd);
               /* 释放线程缓冲区 */
               pthread_mutex_lock(&buff_mutex);
               thread_buff[buffer_index].tid=0;
               thread_buff[buffer_index].buff_status=BUFF_FREED;
               pthread_mutex_unlock(&buff_mutex);
            }
         }
      }

      MainWindow::MainWindow(QWidget *parent)
         : QMainWindow(parent)
         , ui(new Ui::MainWindow)
      {
         ui->setupUi(this);
         enable_button(isserveropened);        // 按键使能
         // 设置工具栏图标样式
         ui->toolBar->setToolButtonStyle(Qt::ToolButtonTextUnderIcon);
         QTimer *timer = new QTimer();       // 设置 1 秒定时器
         connect(timer, SIGNAL(timeout()), this, SLOT(update()));
         timer->start(1000);
      }

      MainWindow::~MainWindow()
      {
         delete ui;
      }

      /* 更新界面缓冲区内容,每秒执行一次 */
      void MainWindow::paintEvent(QPaintEvent*){
         int i;
         // qDebug() << "Here!!!";
         pthread_mutex_lock(&info_mutex);
         for(i=0;i<INFO_NUM;i++){
            if(info[i].status==INFO_OCCUPIED) {
               display_info(info[i].msg);
               info[i].status=INFO_FREED;
```

```
            }
        }
        pthread_mutex_unlock(&info_mutex);
        return;
    }

    /* 向界面输出 msg 消息 */
    void MainWindow::display_info(QString msg){
        ui->textBrowser->append(msg);
    }
    /* 按钮使能函数 */
    void MainWindow::enable_button(bool boolean){
        // 客户端操作
        ui->action_start->setEnabled(!boolean);
        ui->action_stop->setEnabled(boolean);
        // 机票查询
        ui->action_inquireone->setEnabled(boolean);
        ui->action_inquireall->setEnabled(boolean);
    }
    void MainWindow::on_action_start_triggered(){
        ui->textBrowser->append("Server Start...");
        char msg[512];                  //提示信息
        /* 初始化数据结构 */
        init_thread_buff();
        read_ticket_list();
        if(!isserveropened){
            /* 创建套接字 */
            listen_fd=socket(AF_INET,SOCK_STREAM,0);
            if(listen_fd<0) {
                sprintf(msg,"创建监听套接字出错！ \n");
                display_info(msg);
                return;
            }
            /* 填写服务器的地址信息 */
            server.sin_family=AF_INET;
            server.sin_addr.s_addr=htonl(INADDR_ANY);
            server.sin_port=htons(SERVER_PORT_NO);
            setsockopt(listen_fd,SOL_SOCKET,SO_REUSEADDR,(void
*)&flag,sizeof(int));
            /* 绑定端口 */
            ret=bind(listen_fd,(struct sockaddr*)&server, sizeof(server));
            if(ret<0) {
                sprintf(msg,"绑定 TCP 端口：%d 出错！ \n",SERVER_PORT_NO);
                display_info(msg);
                ::close(listen_fd);
```

```
            return;
        }
        /* 转化成倾听套接字 */
        listen(listen_fd,5);
        ret=pthread_create(&listentid, NULL, listen_thread, NULL);
        if(ret==-1) {
            sprintf(msg,"创建监听线程出错！ \n");
            display_info(msg);
            ::close(listen_fd);
            return;
        }
        /* 成功后输出提示信息 */
        sprintf(msg,"服务器端开启成功！服务器在端口：%d 准备接受连接！
\n",SERVER_PORT_NO);
        display_info(msg);
        isserveropened=true;
        enable_button(isserveropened);
    }
}

void MainWindow::on_action_stop_triggered(){
    ui->textBrowser->append("Server Stop...");
    char msg[512];
    if(isserveropened){
        pthread_mutex_lock(&buff_mutex);
        for(i=0;i<THREAD_BUFF_NUM;i++) {
            if(thread_buff[i].buff_status!=BUFF_FREED) {
                /* 退出服务线程 */
                pthread_cancel(thread_buff[i].tid);
                pthread_join(thread_buff[i].tid,NULL);
                /* 退出的线程不释放它的缓冲区，释放工作由主线程来处理 */
                thread_buff[i].tid=0;
                thread_buff[i].buff_status=BUFF_FREED;
                ::close(thread_buff[i].conn_fd);
            }
        }
        pthread_mutex_unlock(&buff_mutex);
        pthread_cancel(listentid);
        pthread_join(listentid,NULL);
        ::close(listen_fd);
        sprintf(msg,"服务器端成功关闭！ \n");
        display_info(msg);
        isserveropened=false;
        enable_button(isserveropened);
    }
}
```

```
void MainWindow::on_action_exit_triggered(){
    ui->textBrowser->append("server Exit...");
    char msg[512];
    if(isserveropened) {
        ::close(listen_fd);
        sprintf(msg,"断开连接成功！\n");
        display_info(msg);
        isserveropened=false;
    }
    enable_button(isserveropened);
    close();
}

void MainWindow::on_action_inquireone_triggered(){
    ui->textBrowser->append("Inquire One...");
    QDialog dialog(this);
    QFormLayout form(&dialog);
    dialog.setWindowTitle("机票查询");
    QList<QLineEdit *> fields;
    QLineEdit *ord = new QLineEdit(&dialog);
    form.addRow(new QLabel("请输入要查询的航班号:"));
    form.addRow(ord);
    fields << ord;
    QDialogButtonBox buttonBox(QDialogButtonBox::Ok |
QDialogButtonBox::Cancel, Qt::Horizontal, &dialog);
    form.addRow(&buttonBox);
    QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
    QObject::connect(&buttonBox, SIGNAL(rejected()), &dialog, SLOT(reject()));

    /* 点击确认按钮 */
    if (dialog.exec() == QDialog::Accepted){
        char msg[512];
        read_ticket_list();    // 读取数据库机票信息
        update_ticket_number();
        QString flight_ord = ord->text();
        unsigned int flight_ID = flight_ord.toInt();
        if(flight_ID<=0 || flight_ID>(unsigned int)numRows) {   //判断输入的航班号
是否正确，不正确的话，给出提示信息，重新输入。
            display_info("输入的航班号错误！请重新输入！");
            return;
        }
        for(i=0;i<numRows;i++) {
            pthread_mutex_lock(&ticket_list[i].ticket_mutex);
            if(ticket_list[i].flight_ID==flight_ID) {
```

```
            sprintf(msg,"你查询的航班号是：%d, 剩余票数：%d,票价：%d\n",ticket_list[i].flight_ID,ticket_list[i].ticket_num,ticket_list[i].ticket_price);
            display_info(msg);
            pthread_mutex_unlock(&ticket_list[i].ticket_mutex);
            break;
          }
          pthread_mutex_unlock(&ticket_list[i].ticket_mutex);
        }
      }
    }

    void MainWindow::on_action_inquireall_triggered(){
      ui->textBrowser->append("Inquire All...");
      int i;
      char msg[512];
      read_ticket_list();    // 读取数据库机票信息
      update_ticket_number();

      for(i=0;i<numRows;i++) {
        sprintf(msg,"航班号：%d, 剩余票数：%d, 票价：%d",\
        ticket_list[i].flight_ID,ticket_list[i].ticket_num, ticket_list[i].ticket_price);
        display_info(msg);
      }
      display_info("\n");
    }

    void MainWindow::on_action_show_triggered(){
      ui->textBrowser->append("Show Message...");
      QDialog dialog(this);
      QFormLayout form(&dialog);
      dialog.setWindowTitle("帮助信息");
      form.addRow(new QLabel("<h1><center>功能说明</center></h1>"));
      form.addRow(new QLabel("开启服务器: 启动服务器程序"));
      form.addRow(new QLabel("关闭服务器: 关闭服务器程序"));
      form.addRow(new QLabel("购买机票: 购买机票"));
      form.addRow(new QLabel("特定航班查询: 查询某一特定航班机票信息"));
      form.addRow(new QLabel("所有航班查询: 查询所有航班机票信息"));
      QDialogButtonBox buttonBox(QDialogButtonBox::Ok, &dialog);
      form.addRow(&buttonBox);
      QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
      if (dialog.exec() == QDialog::Accepted) {
        display_info("查询信息成功\n");
      }
    }
```

```
void MainWindow::on_action_about_triggered(){
    ui->textBrowser->append("Show About...");
    QDialog dialog(this);
    QFormLayout form(&dialog);
    dialog.setWindowTitle("关于");
    form.addRow(new QLabel("<h1>网络售票模拟系统服务端</h1>"));
    form.addRow(new QLabel("<center>版本 V0.3</center>"));
    form.addRow(new QLabel("本程序仅用于测试，请勿用于商业目的"));
    form.addRow(new QLabel("作者信息: 孙硕、戚莘凯、张厚今"));
    form.addRow(new QLabel("更新日期: 2020 年 06 月 17 日"));
    QDialogButtonBox buttonBox(QDialogButtonBox::Ok, &dialog);
    form.addRow(&buttonBox);
    QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
    if (dialog.exec() == QDialog::Accepted) {
        display_info("查询信息成功\n");
    }
}
```

2.2 TicketingSystem_Sell 售票端代码

（1）TicketingSystem_Sell.pro 文件

```
QT    += core gui
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
CONFIG += c++11
# The following define makes your compiler emit warnings if you use
# any Qt feature that has been marked deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS
# You can also make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all
the APIs deprecated before Qt 6.0.0

SOURCES += \
    login.cpp \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    globalapi.h \
    login.h \
    mainwindow.h \
    ticket.h
```

```
FORMS += \
    login.ui \
    mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

RESOURCES += \
    Icon.qrc

LIBS += \
    -lmysqlclient
```

（2）globalapi.h 文件

```
#ifndef    __GLOBALAPI_H
#define    __GLOBALAPI_H
#include <sys/types.h>            /*基本的系统数据类型*/
#include <sys/socket.h>           /*基本的套接字的定义*/
#include <sys/time.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/un.h>
#include <string.h>
#include <pthread.h>
#include <mysql/mysql.h>

/*服务器端使用的端口*/
#define    SERVER_PORT_NO 8889

/*客户端与服务器端使用的消息类型定义*/
#define    INITIAL_VALUE     65535

/*客户端使用的消息代码含义*/
#define    DISCONNECT        0
#define    BUY_TICKET        1
#define    INQUIRE_ONE       2
#define    INQUIRE_ALL       3
```

```
#define ADD_TICKET  4
#define UPDATE_TICKET 5
#define DELETE_TICKET 6

/*服务器端使用的消息代码含义*/
#define    BUY_SUCCEED              255
#define    BUY_FAILED               256
#define    INQUIRE_SUCCEED          257
#define    UNKNOWN_CODE  258

/*服务器端的线程缓冲区的最大数量*/
#define    THREAD_BUFF_NUM          128
/*提示信息输出*/
#define    INFO_NUM                 10
#define    INFO_OCCUPIED            1
#define    INFO_FREED               0
pthread_mutex_t          info_mutex;

/*************message 是客户端与服务器之间的消息结构体*************/
/*服务器与客户端使用的消息结构定义，用来向服务器请求不同类型的信息*/
struct stMessage {
   unsigned int msg_type;              // 用来向服务器请求不同类型的信息
   unsigned int flight_ID;             // 航班号
   unsigned int ticket_num;            // 机票张数
   unsigned int ticket_total_price;    // 机票价钱
} message;

/* 将消息数据类型进行初始化 */
void init_message(){
   message.msg_type=INITIAL_VALUE;
   message.flight_ID=0;
   message.ticket_num=0;
   message.ticket_total_price=0;
}
#endif
```

（3）login.h 文件

```
#ifndef LOGIN_H
#define LOGIN_H
#include <QDialog>

namespace Ui {
   class Login;
}
```

```
class Login : public QDialog
{
    Q_OBJECT

public:
    explicit Login(QWidget *parent = nullptr);
    ~Login();

private slots:
    void on_pushButton_clicked();

private:
    Ui::Login *ui;
};

#endif // LOGIN_H
```

（4）mainwindow.h 文件

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QDialog>
#include <QFormLayout>
#include <QLineEdit>
#include <QDialogButtonBox>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    void display_info(QString msg);
    void enable_button(bool boolean);
private slots:
    void on_action_connect_triggered();
    void on_action_disconnect_triggered();
    void on_action_buyticket_triggered();
    void on_action_exit_triggered();
    void on_action_inquireone_triggered();
    void on_action_inquireall_triggered();
```

```
    void on_action_show_triggered();
    void on_action_about_triggered();
    void on_action_add_triggered();
    void on_action_update_triggered();
    void on_action_delete_triggered();

private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

（5）ticket.h 文件

```
/*ticket.h*/
#ifndef    __TICKET_H
#define    __TICKET_H
#include "globalapi.h"

#define    FLIGHT_NUM 10              // 航班总数
int numRows = 0;

/* 机票的一个简单描述 */
typedef struct ticket_struct_t {
    unsigned int flight_ID;           // 航班号
    unsigned int ticket_num;          // 机票剩余票数
    unsigned int ticket_price;        // 票价
    // 多个线程操作时，必须对机票的剩余数量进行保护
    // 应当对每一个 ticket_num 使用不同的互斥锁
    // 否则将对线程间并行性有较大影响
    pthread_mutex_t        ticket_mutex;
} ticket_struct;
ticket_struct ticket_list[FLIGHT_NUM];
/* read_ticket_list:初始化 ticket_list 数组 */
void read_ticket_list(){
    MYSQL mysql;
    MYSQL_RES * result;
    MYSQL_ROW row;
    int i = 0;
    mysql_init(&mysql);
    mysql_real_connect(&mysql, "localhost", "zhj", "666588", "linux", 0, NULL, 0);
    // 调用 mysql_store_result 之前必须检索数据库
    mysql_query(&mysql, "select * from tickets");
    result = mysql_store_result(&mysql);          // 将查询的全部结果读取到客户端
    numRows = mysql_num_rows(result);         // 统计结果集的行数
    if(result){
```

```
        for(i=0;i<numRows;i++){
            if((row = mysql_fetch_row(result)) != NULL){
                ticket_list[i].flight_ID = atoi(row[0]);
                ticket_list[i].ticket_num = atoi(row[1]);
                ticket_list[i].ticket_price = atoi(row[2]);
            }
        }
    }
    mysql_free_result(result);                  // 释放 result 空间，避免内存泄漏
    mysql_close(&mysql);
    return;
}

void update_ticket_number(void){
    MYSQL mysql;
    MYSQL_RES * result;
    mysql_init(&mysql);
    mysql_real_connect(&mysql, "localhost", "zhj", "666588", "linux", 0, NULL, 0);
    mysql_query(&mysql, "select * from tickets");
    result = mysql_store_result(&mysql);        // 将查询的全部结果读取到客户端
    numRows = mysql_num_rows(result);           // 统计结果集的行数
    mysql_free_result(result);
    mysql_close(&mysql);
}

#endif
```

（6）login.cpp 文件

```
#include "login.h"
#include "ui_login.h"
#include<QMessageBox>

Login::Login(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Login)
{
    ui->setupUi(this);
}

Login::~Login()
{
    delete ui;
}

void Login::on_pushButton_clicked()
```

```
{
   if((ui->lineEdit->text()=="qxk"&&ui->lineEdit_2->text()=="qixinkai")||\
      (ui->lineEdit->text()=="ss"&&ui->lineEdit_2->text()=="sunshuo")||\
      (ui->lineEdit->text()=="zhj"&&ui->lineEdit_2->text()=="zhanghoujin")){
      accept();
   }
   else{
      QMessageBox::warning(this,tr("warning"),\
      tr("user name or password error!"),QMessageBox::Yes);
   }
}
```

（7）main.cpp 文件

```
#include "mainwindow.h"
#include "login.h"
#include <QApplication>

int main(int argc, char *argv[])
{
   QApplication a(argc, argv);
   MainWindow w;
   Login d1;
   d1.setWindowTitle("登录界面");
   if(d1.exec()==QDialog::Accepted)
   {
      w.show();
   }
   return a.exec();
}
```

（8）mainwindow.cpp 文件

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "QDebug"
#include "globalapi.h"
#include "ticket.h"

int ret = 0;                        // 连接状态返回值
int socket_fd;                      // 套接字描述符
struct sockaddr_in server;          // 服务器地址信息，客户端地址信息
int isconnected = false;            // 是否已连接服务器

MainWindow::MainWindow(QWidget *parent)
   : QMainWindow(parent)
```

```
    , ui(new Ui::MainWindow){
    ui->setupUi(this);
    enable_button(isconnected);
    // 设置工具栏图标样式
    ui->toolBar->setToolButtonStyle(Qt::ToolButtonTextUnderIcon);
}

MainWindow::~MainWindow(){
    delete ui;
}

void MainWindow::display_info(QString msg){
    ui->textBrowser->append(msg);
}

void MainWindow::enable_button(bool boolean){
    // 客户端操作
    ui->action_connect->setEnabled(!boolean);
    ui->action_disconnect->setEnabled(boolean);
    ui->action_buyticket->setEnabled(boolean);
    // 机票查询
    ui->action_inquireone->setEnabled(boolean);
    ui->action_inquireall->setEnabled(boolean);
    // 管理员操作
    ui->action_add->setEnabled(boolean);
    ui->action_update->setEnabled(boolean);
    ui->action_delete->setEnabled(boolean);
}

void MainWindow::on_action_connect_triggered(){
    ui->textBrowser->append("Connecting Server....\n");
    char msg[512];                      // 提示信息

    if(!isconnected){
        /* 创建套接字 */
        socket_fd=socket(AF_INET,SOCK_STREAM,0);
        if(socket_fd<0) {
            sprintf(msg,"创建套接字出错！  \n");
            display_info(msg);
            return;
        }

        /* 设置接收、发送超时值 */
        struct timeval time_out;
        time_out.tv_sec=5;
```

```
        time_out.tv_usec=0;
        setsockopt(socket_fd, SOL_SOCKET, SO_RCVTIMEO,\
        &time_out, sizeof(time_out));

        /* 填写服务器的地址信息 */
        server.sin_family=AF_INET;
        server.sin_addr.s_addr=inet_addr("127.0.0.1");//htonl(INADDR_ANY);
        server.sin_port=htons(SERVER_PORT_NO);

        /* 连接服务器 */
        ret = ::connect(socket_fd,(struct sockaddr*)&server, sizeof(server));
        if(ret<0)
        {
           // 这里改了一下，添加了格式控制符%d
           sprintf(msg,"连接服务器出错！%d\n",SERVER_PORT_NO);
           display_info(msg);
           ::close(socket_fd);
           return;
        }

        /* 成功后输出提示信息 */
        sprintf(msg,"连接服务器成功！\n");
        display_info(msg);
        isconnected=true;
        enable_button(isconnected);
    }
}

void MainWindow::on_action_disconnect_triggered(){
    ui->textBrowser->append("Disonnect Server....\n");
    char msg[512];
    if(isconnected) {
        ::close(socket_fd);
        sprintf(msg,"断开连接成功！\n");
        display_info(msg);
        isconnected=false;
    }
    enable_button(isconnected);
}

void MainWindow::on_action_buyticket_triggered(){
    ui->textBrowser->append("Buy Ticket....\n");
//    char* ticket_num_temp = nullptr;
    QDialog dialog(this);
    QFormLayout form(&dialog);
```

```
    dialog.setWindowTitle("机票购买");
    QList<QLineEdit *> fields;
    QLineEdit *ord = new QLineEdit(&dialog);
//    update_ticket_number();
//    sprintf(ticket_num_temp, "请输入要查询的航班号(1-%d):", numRows);
//    form.addRow(new QLabel(ticket_num_temp));
    form.addRow(new QLabel("请输入要查询的航班号:"));
    form.addRow(ord);
    QLineEdit *cnt = new QLineEdit(&dialog);
    form.addRow(new QLabel("请输入要购买票的张数:"));
    form.addRow(cnt);
    fields << ord;
    fields << cnt;
    QDialogButtonBox buttonBox(QDialogButtonBox::Ok |\
    QDialogButtonBox::Cancel, Qt::Horizontal, &dialog);
    form.addRow(&buttonBox);
    QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
    QObject::connect(&buttonBox, SIGNAL(rejected()), &dialog, SLOT(reject()));

    /* 点击确认按钮 */
    if (dialog.exec() == QDialog::Accepted) {
        QString flight_ord = ord->text();
        QString flight_cnt = cnt->text();
        /* 获取输入的航班号 */
        char msg[512];
        char send_buf[512],recv_buf[512];
        int flight_ID = flight_ord.toInt();
        int ticket_num= flight_cnt.toInt();
        update_ticket_number();
        /* 判断输入的航班号是否正确，不正确的话，给出提示信息重新输入 */
        if(flight_ID<=0 || flight_ID>numRows) {
            display_info("输入的航班号错误！请重新输入！");
            return;
        }
        /* 判断输入的票数是否正确，不正确的话，给出提示信息，重新输入 */
        if(ticket_num<=0) {
            display_info("输入的票数错误！请重新输入！");
            return;
        }
        /* 购买机票 */
        init_message();
        message.msg_type=BUY_TICKET;
        message.flight_ID=flight_ID;
        message.ticket_num=ticket_num;
        memcpy(send_buf,&message,sizeof(message));
```

```
            int ret=send(socket_fd, send_buf,sizeof(message),0);
            /* 发送出错 */
            if(ret == -1) {
               display_info("发送失败！请重新发送！");
               return ;
            }
            ret = recv(socket_fd,recv_buf,sizeof(message),0);
            if(ret==-1) {
               display_info("接收失败！请重新发送！");
               return ;
            }
            memcpy(&message,recv_buf,sizeof(message));
            if(message.msg_type==BUY_SUCCEED){
               sprintf(msg,"购买成功！航班号：%d, 票数：%d, 总票价：%d\n",message.flight_ID, message.ticket_num, message.ticket_total_price);
            }
            else{
               sprintf(msg,"购买失败！航班号：%d, 剩余票数：%d, 请求票数：%d\n",message.flight_ID, message.ticket_num,ticket_num);
            }
            display_info(msg);
         }
      }

      void MainWindow::on_action_exit_triggered(){
         char msg[512];
         ui->textBrowser->append("Exit\n");
         while(isconnected){
            ::close(socket_fd);
            sprintf(msg,"断开连接成功！\n");
            display_info(msg);
            isconnected=false;
         }
         display_info("即将关闭客户端");
         close();
      }

      void MainWindow::on_action_inquireone_triggered(){
         ui->textBrowser->append("Inquire One\n");

      //    char* ticket_num_temp = nullptr;
         QDialog dialog(this);
         QFormLayout form(&dialog);
         dialog.setWindowTitle("机票查询");
         QList<QLineEdit *> fields;
```

```
        QLineEdit *ord = new QLineEdit(&dialog);
    //    update_ticket_number();
    //    sprintf(ticket_num_temp, "请输入要查询的航班号(1-%d):", numRows);
    //    form.addRow(new QLabel(QString(ticket_num_temp)));
        form.addRow(new QLabel("请输入要查询的航班号:"));
        form.addRow(ord);
        fields << ord;
        QDialogButtonBox buttonBox(QDialogButtonBox::Ok |
QDialogButtonBox::Cancel, Qt::Horizontal, &dialog);
        form.addRow(&buttonBox);
        QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
        QObject::connect(&buttonBox, SIGNAL(rejected()), &dialog, SLOT(reject()));

        /* 点击确认按钮 */
        if (dialog.exec() == QDialog::Accepted) {
            QString flight_ord = ord->text();
            /* 获取输入的航班号 */
            char msg[512];
            char send_buf[512],recv_buf[512];
            int flight_ID = flight_ord.toInt();
            update_ticket_number();
            // 判断输入的航班号是否正确，不正确的话，给出提示信息，重新输入。
            if(flight_ID<=0 || flight_ID>numRows) {
                display_info("输入的航班号错误！请重新输入！");
                return;
            }
            init_message();
            message.msg_type=INQUIRE_ONE;
            message.flight_ID=flight_ID;
            memcpy(send_buf,&message,sizeof(message));
            int ret=send(socket_fd, send_buf,sizeof(message),0);

            /* 发送出错 */
            if(ret==-1) {
                display_info("发送失败！请重新发送！");
                return ;
            }
            ret=recv(socket_fd,recv_buf,sizeof(message),0);
            if(ret==-1) {
                display_info("接收失败！请重新连接服务器！\n");
                return ;
            }

            memcpy(&message,recv_buf,sizeof(message));
            if(message.msg_type==INQUIRE_SUCCEED){
```

```
            sprintf(msg,"查询成功！航班号：%d, 剩余票数：%d, 票价：%d\n",
message.flight_ID, message.ticket_num,  message.ticket_total_price);
            }
            else{
              sprintf(msg,"查 询 失 败 ！ 航 班 号 ：%d, 剩 余 票 数 ： 未 知
\n",message.flight_ID);
            }
            display_info(msg);
          }
        }

        void MainWindow::on_action_inquireall_triggered(){
          ui->textBrowser->append("Inquire All\n");
          int i,pos;
          char msg[512];
          char send_buf[1024], recv_buf[1024];
          init_message();
          message.msg_type=INQUIRE_ALL;
          memcpy(send_buf,&message,sizeof(message));
          int ret=send(socket_fd, send_buf,sizeof(message),0);
          /* 发送出错 */
          if(ret==-1) {
            display_info("发送失败！请重新发送！");
            return;
          }
          ret=recv(socket_fd,recv_buf,sizeof(recv_buf),0);
          if(ret==-1) {
            display_info("接收失败！请重新发送！");
            return;
          }
          pos=0;
          sprintf(msg,"查询所有航班结果：\n");
          display_info(msg);
          for (i=0;i<ret;i=i+sizeof(message)) {
            memcpy(&message,recv_buf+pos,sizeof(message));
            if(message.msg_type==INQUIRE_SUCCEED){
              sprintf(msg,"查 询 成 功 ！ 航 班 号 ：%d, 剩 余 票 数 ：%d, 票
价：%d",message.flight_ID, message.ticket_num, message.ticket_total_price);
            }
            else{
              sprintf(msg,"查 询 失 败 ！ 航 班 号 ：%d, 剩 余 票 数 ： 未 知
",message.flight_ID);
            }
            display_info(msg);
            pos+=sizeof(message);
```

```
    }
    display_info("\n");
}

void MainWindow::on_action_show_triggered(){
    ui->textBrowser->append("Show User Manual....\n");

    QDialog dialog(this);
    QFormLayout form(&dialog);
    dialog.setWindowTitle("帮助信息");
    form.addRow(new QLabel("<h1><center>功能说明</center></h1>"));
    form.addRow(new QLabel("连接服务器: 与远程服务器建立连接"));
    form.addRow(new QLabel("断开连接：断开与远程服务器的连接"));
    form.addRow(new QLabel("购买机票：购买机票"));
    form.addRow(new QLabel("特定航班查询：查询某一特定航班机票信息"));
    form.addRow(new QLabel("所有航班查询：查询所有航班机票信息"));
    form.addRow(new QLabel("增加航班信息：增加指定的航班机票信息"));
    form.addRow(new QLabel("更新航班信息：更新指定的航班机票信息"));
    form.addRow(new QLabel("删除航班信息：删除指定的航班机票信息"));
    QDialogButtonBox buttonBox(QDialogButtonBox::Ok, &dialog);
    form.addRow(&buttonBox);
    QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
    if (dialog.exec() == QDialog::Accepted) {
        display_info("查询信息成功\n");
    }
}

void MainWindow::on_action_about_triggered(){
    ui->textBrowser->append("About\n");
    QDialog dialog(this);
    QFormLayout form(&dialog);
    dialog.setWindowTitle("关于");
    form.addRow(new QLabel("<h1>网络售票模拟系统管理端</h1>"));
    form.addRow(new QLabel("<center>版本 V0.3</center>"));
    form.addRow(new QLabel("本程序仅用于测试，请勿用于商业目的"));
    form.addRow(new QLabel("作者信息: 孙硕、戚莘凯、张厚今"));
    form.addRow(new QLabel("更新日期: 2020 年 06 月 17 日"));
    QDialogButtonBox buttonBox(QDialogButtonBox::Ok, &dialog);
    form.addRow(&buttonBox);
    QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
    if (dialog.exec() == QDialog::Accepted) {
        display_info("查询信息成功\n");
    }
}
```

```
void MainWindow::on_action_add_triggered()
{
    ui->textBrowser->append("Add Information...");

    QDialog dialog(this);
    QFormLayout form(&dialog);
    dialog.setWindowTitle("增加航班信息(管理员)");
    QList<QLineEdit *> fields;
    QLineEdit *id = new QLineEdit(&dialog);
    form.addRow(new QLabel("请输入该航班的航班号:"));
    form.addRow(id);
    QLineEdit *number = new QLineEdit(&dialog);
    form.addRow(new QLabel("请输入该航班的票数:"));
    form.addRow(number);
    QLineEdit *price = new QLineEdit(&dialog);
    form.addRow(new QLabel("请输入该航班的票价:"));
    form.addRow(price);
    fields << id;
    fields << number;
    fields << price;
    QDialogButtonBox buttonBox(QDialogButtonBox::Ok |
QDialogButtonBox::Cancel, Qt::Horizontal, &dialog);
    form.addRow(&buttonBox);
    QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
    QObject::connect(&buttonBox, SIGNAL(rejected()), &dialog, SLOT(reject()));

    if (dialog.exec() == QDialog::Accepted) {
        int flight_ID = id->text().toInt();
        int ticket_num = number->text().toInt();
        int ticket_price = price->text().toInt();
        char send_buf[1024];
        init_message();
        message.msg_type=ADD_TICKET;
        message.flight_ID = flight_ID;
        message.ticket_num = ticket_num;
        message.ticket_total_price = ticket_price;
        memcpy(send_buf,&message,sizeof(message));

        int ret=send(socket_fd, send_buf,sizeof(message),0);
        /* 发送出错 */
        if(ret==-1) {
            display_info("发送失败！请重新发送！");
            return;
        }
        display_info("增加航班信息成功!\n");
```

```
    }
}

void MainWindow::on_action_update_triggered()
{
    ui->textBrowser->append("Update Information...");

    QDialog dialog(this);
    QFormLayout form(&dialog);
    dialog.setWindowTitle("更新航班信息(管理员)");
    QList<QLineEdit *> fields;
    QLineEdit *id = new QLineEdit(&dialog);
    form.addRow(new QLabel("请输入该航班的航班号:"));
    form.addRow(id);
    QLineEdit *number = new QLineEdit(&dialog);
    form.addRow(new QLabel("请输入该航班的票数:"));
    form.addRow(number);
    QLineEdit *price = new QLineEdit(&dialog);
    form.addRow(new QLabel("请输入该航班的票价:"));
    form.addRow(price);
    fields << id;
    fields << number;
    fields << price;
    QDialogButtonBox buttonBox(QDialogButtonBox::Ok |
QDialogButtonBox::Cancel, Qt::Horizontal, &dialog);
    form.addRow(&buttonBox);
    QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
    QObject::connect(&buttonBox, SIGNAL(rejected()), &dialog, SLOT(reject()));

    if (dialog.exec() == QDialog::Accepted) {
        int flight_ID = id->text().toInt();
        int ticket_num = number->text().toInt();
        int ticket_price = price->text().toInt();

        char send_buf[1024];
        init_message();
        message.msg_type=UPDATE_TICKET;
        message.flight_ID = flight_ID;
        message.ticket_num = ticket_num;
        message.ticket_total_price = ticket_price;
        memcpy(send_buf,&message,sizeof(message));

        int ret=send(socket_fd, send_buf,sizeof(message),0);
        /* 发送出错 */
        if(ret==-1) {
            display_info("发送失败！请重新发送！");
```

```
            return;
        }
        display_info("更新航班信息成功!\n");
    }

}

void MainWindow::on_action_delete_triggered()
{
    ui->textBrowser->append("Delete Information...");

    QDialog dialog(this);
    QFormLayout form(&dialog);
    dialog.setWindowTitle("删除航班信息(管理员)");
    QList<QLineEdit *> fields;
    QLineEdit *id = new QLineEdit(&dialog);
    form.addRow(new QLabel("请输入该航班的航班号:"));
    form.addRow(id);
    fields << id;
    QDialogButtonBox buttonBox(QDialogButtonBox::Ok |
QDialogButtonBox::Cancel, Qt::Horizontal, &dialog);
    form.addRow(&buttonBox);
    QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
    QObject::connect(&buttonBox, SIGNAL(rejected()), &dialog, SLOT(reject()));

    if (dialog.exec() == QDialog::Accepted) {
        int flight_ID = id->text().toInt();
        char send_buf[1024];
        init_message();
        message.msg_type=DELETE_TICKET;
        message.flight_ID = flight_ID;
        memcpy(send_buf,&message,sizeof(message));

        int ret=send(socket_fd, send_buf,sizeof(message),0);
        /* 发送出错 */
        if(ret==-1) {
            display_info("发送失败！请重新发送！");
            return;
        }
        display_info("删除航班信息成功!\n");
    }
}
```

2.3 TicketingSystem_Client 购票端代码

（1）TicketingSystem_Client.pro 文件

```
QT   += core gui
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
CONFIG += c++11

# The following define makes your compiler emit warnings if you use
# any Qt feature that has been marked deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS
# You can also make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all
the APIs deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    mainwindow.cpp \
    welcome.cpp

HEADERS += \
    globalapi.h \
    mainwindow.h \
    ticket.h \
    welcome.h

FORMS += \
    mainwindow.ui \
    welcome.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
RESOURCES += \
    Icon.qrc

LIBS += \
    -lmysqlclient
```

（2）globalapi.h

```
#ifndef    __GLOBALAPI_H
#define    __GLOBALAPI_H
#include <sys/types.h>           /*基本的系统数据类型*/
#include <sys/socket.h>          /*基本的套接字的定义*/
#include <sys/time.h>
```

```
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/un.h>
#include <string.h>
#include <pthread.h>
#include <mysql/mysql.h>

/*服务器端使用的端口*/
#define    SERVER_PORT_NO          8889

/*客户端与服务器端使用的消息类型定义*/
#define    INITIAL_VALUE           65535

/*客户端使用的消息代码含义*/
#define    DISCONNECT              0
#define    BUY_TICKET              1
#define    INQUIRE_ONE             2
#define    INQUIRE_ALL             3

/*服务器端使用的消息代码含义*/
#define    BUY_SUCCEED             255
#define    BUY_FAILED              256
#define    INQUIRE_SUCCEED         257
#define    UNKNOWN_CODE            258

/*服务器端的线程缓冲区的最大数量*/
#define    THREAD_BUFF_NUM 128

/*提示信息输出*/
#define    INFO_NUM                10
#define    INFO_OCCUPIED           1
#define    INFO_FREED              0

pthread_mutex_t          info_mutex;

/*************message 是客户端与服务器之间的消息结构体**************/
/*服务器与客户端使用的消息结构定义，用来向服务器请求不同类型的信息*/
struct stMessage {
   //客户端可以取值为 DISCONNECT：断开连接；BUY_TICKET：购买机票
   // INQUIRE_ONE：查询特定航班机票；INQUIRE_ALL：查询所有航班机票
```

```
    unsigned int    msg_type;           // 用来向服务器请求不同类型的消息
    unsigned int    flight_ID;          // 航班号
    unsigned int    ticket_num;         // 机票张数
    unsigned int    ticket_total_price; // 机票价钱
} message;

/* 将消息数据类型进行初始化 */
void init_message()
{
    message.msg_type=INITIAL_VALUE;
    message.flight_ID=0;
    message.ticket_num=0;
    message.ticket_total_price=0;
}

/*********以下是有关消息输出缓冲区的操作函数*************/
struct info_t {
    int     status;         /*INFO_OCCUPIED or INFO_FREED 消息缓冲区的状
态*/
    char    msg[512];       /*contents of message 服务器返回的具体消息内容*/
} info[INFO_NUM];

/*初始化界面输出信息缓冲区*/
void init_info(){
    int i;
    for (i=INFO_NUM;i>0;i--){
            info[i-1].status=INFO_FREED;
    }
    sprintf(info[i-1].msg," ");
}

/*分配一个空闲的界面输出信息缓冲区，如果没有空闲的缓冲区则返回-1*/
// 最终会获取空闲的缓冲区下标
int get_free_info(){
    int i,ret;
    /*注意互斥锁的操作，这些操作必须成对的（加锁和解锁），否则会死锁*/
    pthread_mutex_lock(&info_mutex);
    for(i=0;i<INFO_NUM; i++){// 获取当前是空闲状态的缓冲区下标 ret
            if(info[i].status==INFO_FREED) {
                    ret=i;
                    pthread_mutex_unlock(&info_mutex);
                    break;
            }
    }
    if(i==INFO_NUM) {               // 分配空间失败
```

```
            ret=-1;
            pthread_mutex_unlock(&info_mutex);
    }
    return ret;
}

/*释放界面输出信息缓冲区，对 info_status 的访问同样需要使用互斥保护*/
// 释放以 index 为下标的缓冲区
void free_info(int index){
    pthread_mutex_lock(&info_mutex);
    if(info[index].status==INFO_OCCUPIED){
            info[index].status=INFO_FREED;
    }
    pthread_mutex_unlock(&info_mutex);
}

// 为消息缓冲区添加具体的消息内容 src
void add_info(char *src)
{  int i;
    // 获取空闲状态的缓冲区下标
    while((i=get_free_info())==-1){
            sleep(1);
    }
    /*添加消息*/
    pthread_mutex_lock(&info_mutex);
    info[i].status=INFO_OCCUPIED;
    strcpy(info[i].msg, src);
    pthread_mutex_unlock(&info_mutex);
}
#endif
```

（3）mainwindow.h 文件

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QDialog>
#include <QFormLayout>
#include <QLineEdit>
#include <QDialogButtonBox>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
```

```
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    void display_info(QString msg);
    void enable_button(bool boolean);

private slots:
    void on_action_connect_triggered();
    void on_action_disconnect_triggered();
    void on_action_buyticket_triggered();
    void on_action_exit_triggered();
    void on_action_inquireone_triggered();
    void on_action_inquireall_triggered();
    void on_action_show_triggered();
    void on_action_about_triggered();

private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

（4）ticket.h 文件

```
/*ticket.h*/
#ifndef    __TICKET_H
#define    __TICKET_H
#include "globalapi.h"
#define    FLIGHT_NUM 10     //航班总数
int numRows = 0;

/* 机票的简单描述，flight_ID 表示航班号，ticket_num 表示机票剩余票数 */
typedef struct ticket_struct_t {
    unsigned int flight_ID;
    unsigned int ticket_num;
    unsigned int ticket_price;      //票价
    /*多个线程操作时，必须对机票的剩余数量进行保护。由于这样的操作比较频繁，所以应当对每一个 ticket_num 使用不同的互斥锁，否则对线程间并行性有较大影响。*/
    pthread_mutex_t        ticket_mutex;
} ticket_struct;

ticket_struct ticket_list[FLIGHT_NUM];
```

```
/* read_ticket_list:初始化 ticket_list 数组 */
void read_ticket_list(){
    MYSQL mysql;
    MYSQL_RES * result;
    MYSQL_ROW row;
    int i = 0;
    mysql_init(&mysql);
    mysql_real_connect(&mysql, "localhost", "zhj", "666588", "linux", 0, NULL, 0);
    mysql_query(&mysql, "select * from tickets");        // 调用 mysql_store_result 之前必须检索数据库
    result = mysql_store_result(&mysql);          // 将查询的全部结果读取到客户端
    numRows = mysql_num_rows(result);         // 统计结果集的行数
    if(result){
        for(i=0;i<numRows;i++){
            if((row = mysql_fetch_row(result)) != NULL){
                ticket_list[i].flight_ID = atoi(row[0]);
                ticket_list[i].ticket_num = atoi(row[1]);
                ticket_list[i].ticket_price = atoi(row[2]);
            }
        }
    }
    mysql_free_result(result);//释放 result 空间，避免内存泄漏
    mysql_close(&mysql);
    return;
}

void update_ticket_number(void){
    MYSQL mysql;
    MYSQL_RES * result;
    mysql_init(&mysql);
    mysql_real_connect(&mysql, "localhost", "zhj", "666588", "linux", 0, NULL, 0);
    mysql_query(&mysql, "select * from tickets");
    result = mysql_store_result(&mysql);//将查询的全部结果读取到客户端
    numRows = mysql_num_rows(result);   //统计结果集的行数
    mysql_free_result(result);
    mysql_close(&mysql);
}

#endif
```

（5）welcome.h 文件

```
#ifndef WELCOME_H
#define WELCOME_H
#include <QDialog>
```

```
namespace Ui {
class welcome;
}

class welcome : public QDialog
{
    Q_OBJECT

public:
    explicit welcome(QWidget *parent = nullptr);
    ~welcome();

private slots:
    void on_pushButton_clicked();

private:
    Ui::welcome *ui;
};

#endif // WELCOME_H
```

（6）main.cpp 文件

```
#include "mainwindow.h"
#include "welcome.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    welcome d1;
    d1.setWindowTitle("欢迎界面");
    if(d1.exec()==QDialog::Accepted)
    {
        w.show();
    }
    return a.exec();
}
```

（7）mainwindow.cpp 文件

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "globalapi.h"
#include "ticket.h"
```

```
int ret = 0;                        // 连接状态返回值
int socket_fd;                      // 套接字描述符
struct sockaddr_in server;          // 服务器地址信息，客户端地址信息
int isconnected = false;            // 是否已连接服务器

MainWindow::MainWindow(QWidget *parent)
   : QMainWindow(parent)
   , ui(new Ui::MainWindow){
   ui->setupUi(this);
   enable_button(isconnected);
   // 设置工具栏图标样式
   ui->toolBar->setToolButtonStyle(Qt::ToolButtonTextUnderIcon);
}

MainWindow::~MainWindow(){
   delete ui;
}

void MainWindow::display_info(QString msg){
   ui->textBrowser->append(msg);
}

void MainWindow::enable_button(bool boolean){
   // 客户端操作
   ui->action_connect->setEnabled(!boolean);
   ui->action_disconnect->setEnabled(boolean);
   ui->action_buyticket->setEnabled(boolean);
   // 机票查询
   ui->action_inquireone->setEnabled(boolean);
   ui->action_inquireall->setEnabled(boolean);
}

void MainWindow::on_action_connect_triggered(){
   ui->textBrowser->append("Connecting Server....\n");
   char msg[512];                   // 提示信息

   if(!isconnected){
      /* 创建套接字 */
      socket_fd=socket(AF_INET,SOCK_STREAM,0);
      if(socket_fd<0) {
         sprintf(msg,"创建套接字出错！ \n");
         display_info(msg);
         return;
      }
      /* 设置接收、发送超时值 */
```

```
        struct timeval time_out;
        time_out.tv_sec=5;
        time_out.tv_usec=0;
        setsockopt(socket_fd, SOL_SOCKET, SO_RCVTIMEO,\
        &time_out, sizeof(time_out));

        /* 填写服务器的地址信息 */
        server.sin_family=AF_INET;
        server.sin_addr.s_addr=inet_addr("127.0.0.1");//htonl(INADDR_ANY);
        server.sin_port=htons(SERVER_PORT_NO);

        /* 连接服务器 */
        ret = ::connect(socket_fd,(struct sockaddr*)&server, sizeof(server));
        if(ret<0) {
           // 这里改了一下，添加了格式控制符%d
           sprintf(msg,"连接服务器出错！%d\n",SERVER_PORT_NO);
           display_info(msg);
           ::close(socket_fd);
           return;
        }

        /* 成功后输出提示信息 */
        sprintf(msg,"连接服务器成功！\n");
        display_info(msg);
        isconnected=true;
        enable_button(isconnected);
    }
}

void MainWindow::on_action_disconnect_triggered(){
    ui->textBrowser->append("Disonnect Server....\n");
    char msg[512];
    if(isconnected) {
        ::close(socket_fd);
        sprintf(msg,"断开连接成功！\n");
        display_info(msg);
        isconnected=false;
    }
    enable_button(isconnected);
}

void MainWindow::on_action_buyticket_triggered(){
    ui->textBrowser->append("Buy Ticket....\n");
    QDialog dialog(this);
    QFormLayout form(&dialog);
```

```
        dialog.setWindowTitle("机票购买");
        QList<QLineEdit *> fields;
        QLineEdit *ord = new QLineEdit(&dialog);
        form.addRow(new QLabel("请输入要购买的航班号:"));
        form.addRow(ord);
        QLineEdit *cnt = new QLineEdit(&dialog);
        form.addRow(new QLabel("请输入要购买票的张数:"));
        form.addRow(cnt);
        fields << ord;
        fields << cnt;
        QDialogButtonBox buttonBox(QDialogButtonBox::Ok |
QDialogButtonBox::Cancel, Qt::Horizontal, &dialog);
        form.addRow(&buttonBox);
        QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
        QObject::connect(&buttonBox, SIGNAL(rejected()), &dialog, SLOT(reject()));

        /* 点击确认按钮 */
        if (dialog.exec() == QDialog::Accepted) {
           QString flight_ord = ord->text();
           QString flight_cnt = cnt->text();
           /* 获取输入的航班号 */
           char msg[512];
           char send_buf[512],recv_buf[512];
           int flight_ID = flight_ord.toInt();
           int ticket_num= flight_cnt.toInt();
           update_ticket_number();
           /* 判断输入的航班号是否正确，不正确的话，给出提示信息重新输入 */
           if(flight_ID<=0 || flight_ID>numRows) {
              display_info("输入的航班号错误！请重新输入！");
              return;
           }

           /* 判断输入的票数是否正确，不正确的话，给出提示信息，重新输入 */
           if(ticket_num<=0) {
              display_info("输入的票数错误！请重新输入！");
              return;
           }

           /* 购买机票 */
           init_message();
           message.msg_type=BUY_TICKET;
           message.flight_ID=flight_ID;
           message.ticket_num=ticket_num;
           memcpy(send_buf,&message,sizeof(message));
           int ret=send(socket_fd, send_buf,sizeof(message),0);
```

```
        /* 发送出错 */
        if(ret == -1) {
            display_info("发送失败！请重新发送！");
            return ;
        }
        ret = recv(socket_fd,recv_buf,sizeof(message),0);
        if(ret==-1) {
            display_info("接收失败！请重新发送！");
            return ;
        }
        memcpy(&message,recv_buf,sizeof(message));
        if(message.msg_type==BUY_SUCCEED){
            sprintf(msg,"购买成功！航班号：%d, 票数：%d, 总票价：%d\n",\
                message.flight_ID,message.ticket_num, message.ticket_total_price);
        }
        else{
            sprintf(msg,"购买失败！航班号：%d, 剩余票数：%d, 请求票数：%d\n",\
                message.flight_ID,message.ticket_num,ticket_num);
        }
        display_info(msg);
    }
}

void MainWindow::on_action_exit_triggered(){
    char msg[512];
    ui->textBrowser->append("Exit\n");
    while(isconnected){
        ::close(socket_fd);
        sprintf(msg,"断开连接成功！\n");
        display_info(msg);
        isconnected=false;
    }
    display_info("即将关闭客户端");
    close();
}

void MainWindow::on_action_inquireone_triggered(){
    ui->textBrowser->append("Inquire One\n");
    QDialog dialog(this);
    QFormLayout form(&dialog);
    dialog.setWindowTitle("机票查询");
    QList<QLineEdit *> fields;
    QLineEdit *ord = new QLineEdit(&dialog);
    form.addRow(new QLabel("请输入要查询的航班号:"));
```

```
        form.addRow(ord);
        fields << ord;
        QDialogButtonBox buttonBox(QDialogButtonBox::Ok |
QDialogButtonBox::Cancel, Qt::Horizontal, &dialog);
        form.addRow(&buttonBox);
        QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
        QObject::connect(&buttonBox, SIGNAL(rejected()), &dialog, SLOT(reject()));

        /* 点击确认按钮 */
        if (dialog.exec() == QDialog::Accepted) {
           QString flight_ord = ord->text();
           /* 获取输入的航班号 */
           char msg[512];
           char send_buf[512],recv_buf[512];
           int flight_ID = flight_ord.toInt();
           update_ticket_number();
           // 判断输入的航班号是否正确，不正确的话，给出提示信息，重新输入。
           if(flight_ID<=0 || flight_ID>numRows) {
              display_info("输入的航班号错误！请重新输入！");
              return;
           }
           init_message();
           message.msg_type=INQUIRE_ONE;
           message.flight_ID=flight_ID;
           memcpy(send_buf,&message,sizeof(message));
           int ret=send(socket_fd, send_buf,sizeof(message),0);

           /* 发送出错 */
           if(ret==-1) {
              display_info("发送失败！请重新发送！");
              return ;
           }
           ret=recv(socket_fd,recv_buf,sizeof(message),0);
           if(ret==-1) {
              display_info("接收失败！请重新连接服务器！\n");
              return ;
           }
           memcpy(&message,recv_buf,sizeof(message));
           if(message.msg_type==INQUIRE_SUCCEED){
              sprintf(msg,"查询成功！航班号：%d, 剩余票数：%d, 票价：%d\n",\
                 message.flight_ID,message.ticket_num, message.ticket_total_price);
           }
           else{
              sprintf(msg,"查 询 失 败 ！ 航 班 号 ：%d, 剩 余 票 数 ： 未 知
\n",message.flight_ID);
```

```
            }
            display_info(msg);
        }
    }

    void MainWindow::on_action_inquireall_triggered(){
        ui->textBrowser->append("Inquire All\n");
        int i,pos;
        char msg[512];
        char send_buf[512], recv_buf[512];
        init_message();
        message.msg_type=INQUIRE_ALL;
        memcpy(send_buf,&message,sizeof(message));
        int ret=send(socket_fd, send_buf,sizeof(message),0);

        /* 发送出错 */
        if(ret==-1) {
            display_info("发送失败！请重新发送！") ;
            return ;
        }
        ret=recv(socket_fd,recv_buf,sizeof(recv_buf),0);
        if(ret==-1) {
            display_info("接收失败！请重新发送！") ;
            return ;
        }

        pos=0;
        sprintf(msg,"查询所有航班结果：\n");
        display_info(msg);
        for (i=0;i<ret;i=i+sizeof(message)) {
            memcpy(&message,recv_buf+pos,sizeof(message));
            if(message.msg_type==INQUIRE_SUCCEED){
                sprintf(msg,"查询成功！航班号：%d, 剩余票数：%d, 票价：%d",\
                    message.flight_ID,message.ticket_num, message.ticket_total_price);
            }
            else{
                sprintf(msg,"查 询 失 败 ！ 航 班 号 ：%d, 剩 余 票 数 ： 未 知
",message.flight_ID);
            }
            display_info(msg);
            pos+=sizeof(message);
        }
        display_info("\n");
    }

    void MainWindow::on_action_show_triggered(){
```

```
    ui->textBrowser->append("Show User Manual....\n");
    QDialog dialog(this);
    QFormLayout form(&dialog);
    dialog.setWindowTitle("帮助信息");
    form.addRow(new QLabel("<h1><center>功能说明</center></h1>"));
    form.addRow(new QLabel("连接服务器: 与远程服务器建立连接"));
    form.addRow(new QLabel("断开连接：断开与远程服务器的连接"));
    form.addRow(new QLabel("购买机票：购买机票"));
    form.addRow(new QLabel("特定航班查询：查询某一特定航班机票信息"));
    form.addRow(new QLabel("所有航班查询：查询所有航班机票信息"));
    QDialogButtonBox buttonBox(QDialogButtonBox::Ok, &dialog);
    form.addRow(&buttonBox);
    QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
    if (dialog.exec() == QDialog::Accepted) {
        display_info("查询信息成功\n");
    }
}

void MainWindow::on_action_about_triggered(){
    ui->textBrowser->append("About\n");
    QDialog dialog(this);
    QFormLayout form(&dialog);
    dialog.setWindowTitle("关于");
    form.addRow(new QLabel("<h1>网络售票模拟系统客户端</h1>"));
    form.addRow(new QLabel("<center>版本 V0.3</center>"));
    form.addRow(new QLabel("本程序仅用于测试，请勿用于商业目的"));
    form.addRow(new QLabel("作者信息: 孙硕、戚莘凯、张厚今"));
    form.addRow(new QLabel("更新日期: 2020 年 06 月 17 日"));
    QDialogButtonBox buttonBox(QDialogButtonBox::Ok, &dialog);
    form.addRow(&buttonBox);
    QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
    if (dialog.exec() == QDialog::Accepted) {
        display_info("查询信息成功\n");
    }
}
```

（8）welcome.cpp 文件

```
#include "welcome.h"
#include "ui_welcome.h"

welcome::welcome(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::welcome)
{
```

```
    ui->setupUi(this);
}

welcome::~welcome()
{
    delete ui;
}

void welcome::on_pushButton_clicked()
{
    accept();
}
```