

# 实训报告书

实训题目： QT 编程实训

系（部）： 智能装备学院

专业班级： 电子信息科学与技术 17-2 班

姓 名： 张厚今

学 号： 201723010237

完成日期： 2020 年 6 月 19 日

# 目录

1 实训目的 .....	1
2 实训原理 .....	1
2.1 程序功能分析 .....	1
2.2 界面框架分析 .....	1
3 实训过程 .....	2
3.1 登录界面设计 .....	2
3.2 售票界面 UI 设计 .....	6
3.3 工具栏按键使能 .....	12
3.4 数据输出函数 .....	14
3.5 函数命名空间 .....	14
3.6 信息交互界面 .....	15
4 实现的效果 .....	17
4.1 登录界面及初始化 .....	17
4.2 查询特定航班 .....	19
4.3 查询所有航班 .....	21
4.4 增加航班信息 .....	21
4.5 更新航班信息 .....	24
4.6 删除航班信息 .....	25
4.7 帮助信息和退出程序 .....	27
5 实训中遇到的问题及解决方案 .....	29
6 实训总结 .....	30

实习类型	课程实训	实训地点	学校
组 别		实习课题	飞机票网络售票模拟系统
实训人姓名	张厚今、孙硕、戚莘凯		
任务分配	张厚今：售票端 孙硕：购票端 戚莘凯：服务端		
指导教师	徐文正	实习日期	2020.06.15 至 2020.06.19
实训成绩			
指导教师评语	<div>指导教师签名：_____</div> <div>_____年 ____月 ____日</div>		

## 1 实训目的

本次实训需要设计并实现一个飞机票的网络售票模拟系统，主要包括服务端设计、售票端设计以及购票端设计。在 QT for Linux 环境中移植 Linux C 的底层逻辑代码，在 UI 编辑区添加前端界面，可以实现 Linux 后台与 QT 前端的结合，为以后的项目开发奠定基础。

## 2 实训原理

### 2.1 程序功能分析

首先分析一下整个航班系统的功能需求。航班模拟系统由三部分组成，分别为服务端、售票端和购票端。为方便表述，以下将售票端和购票端统称为客户端。服务端通常是处理性能较好的计算机，适合进行数据处理操作，因此服务端承担了读取航班信息，接收客户端的数据请求以及对航班数据进行相应处理等功能。售票端面向系统管理员，具有较高的管理权限，可以实现对机票的查询、增加、更新和删除等操作。购票端面向普通用户，只有查询航班和购买机票的功能。

因为底层的控制代码在 Linux 实训中已经编写完成了，所以本次 QT 实训只需要为服务端、售票端和购票端分别编写前端 UI 界面即可。另外需要注意的是，购票端是直接面向了普通用户，所以可以为它添加一个简易的欢迎页面。售票端因为具有较高的权限，所以需要增加一个登录界面，只有管理员输入正确的密码之后才能进行接下来的操作。

### 2.2 界面框架分析

前端界面可以使用基于 QMainWindow 的窗口类，这样可以非常方便地添加菜单选项，添加工具栏等各种可视化的图形操作。服务端、售票端和购票端采用统一的 UI 风格，添加菜单栏和工具栏。通过查找开源图库，可以获取相关图标资源，为每个工具栏选项添加图标，方便用户操作。

登录界面和欢迎页面通过新建 QT 设计类来实现。其中登录界面要实现密码判断和界面跳转的逻辑功能。

### 3 实训过程

我们团队设计的网络购票模拟系统，主要由三部分组成，分别是服务端、售票端和购票端。我负责的是售票端部分，下面对售票端的制作过程进行简单介绍。售票端的工程代码结构如下图所示。

在售票端代码中，大部分底层控制逻辑的文件都是直接由 Linux 移植过来的。另外我们还为售票端添加了登录界面等诸多功能。

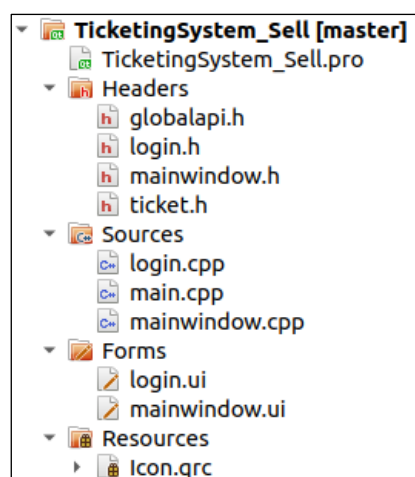


图 3.1 售票端工程结构

售票端的功能有“查询所有航班”、“查询特定航班”、“增加航班信息”、“更新航班信息”和“删除航班信息”等。因为售票端涉及对航班信息的增删改功能，权限较高，因此我们为它添加了一个登录界面，只有管理员输入正确的密码时才能进入售票界面。

#### 3.1 登录界面设计

登录界面的制作首先需要增加一个 QT 设计类。相较于直接在代码中使用 `new QDialog` 语句创建窗口，这种直接添加文件的方式更加灵活，操作也更加简便。QT 设计类可以很方便地生成制作登录界面所需的源文件、头文件和界面文件，选择添加新建文件中的 Qt Designer Form Class，即可创建 QT 模板。如下图 3.2 所示。

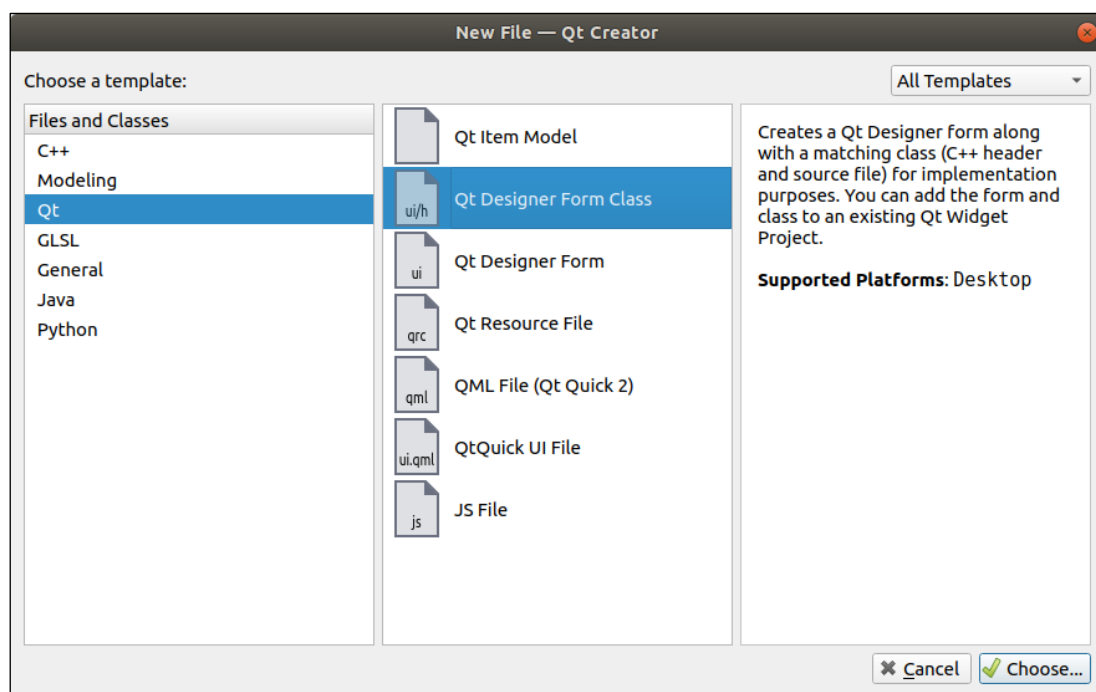


图 3.2 增加设计类

将类名改为 Login，生成对应名称的头文件、源文件和 UI 界面文件。

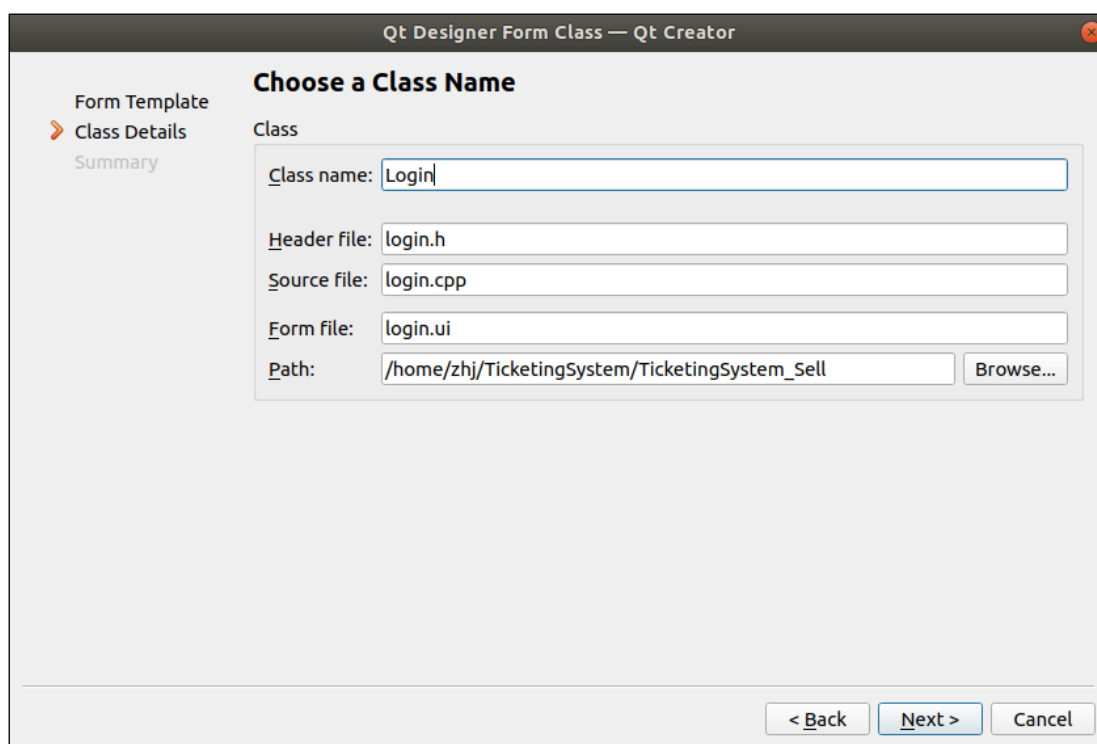


图 3.3 设置类名

首先设计一下 UI 界面，为其添加 Label 标题、LineEdit 输入框、登录按钮等组件。使用水平布局和垂直布局限制界面组件的布局格式，将各个组件有秩序地摆放。设置 Label 标签的 alignment 属性，使之水平垂直居中；设置 font 属性设置字体的大小和字体格式。最终设计的登录界面样式如下图 3.4 所示。

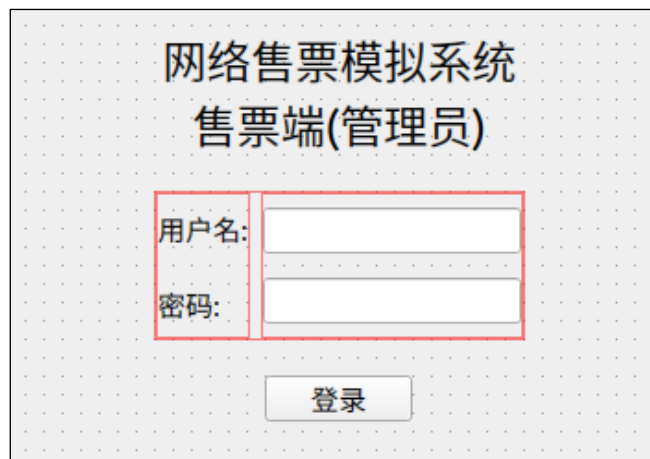


图 3.4 登录界面样式

在 login 的头文件中，需要声明按钮的槽函数，如下所示。

```
public:
    explicit Login(QWidget *parent = nullptr);
    ~Login();
private slots:
    void on_pushButton_clicked();
```

接下来设计登录界面的代码逻辑。当点击“登录”按钮后，需要进入按钮的槽函数进行事件处理。在槽函数中，通过判断管理员输入的密码是否正确，可以做进一步处理。具体代码如下所示。

```
void Login::on_pushButton_clicked()
{
    if((ui->lineEdit->text()=="qzk"&&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);
    }
}
```

在槽函数中，利用“逻辑或”判断如果用户输入的用户名和密码是否正确。代码中设置的三个用户名和密码分别是我们团队三个队员的姓名英文缩写。如果用户输入正确，会执行 accept 函数，否则的话会显示错误信息的提示框，警告用户名或密码输入错误。

接下来需要修改 main.cpp 文件中的内容，设置登录框的标题以及代码逻辑。如下面代码所示，只有当判断到 QDialog::Accepted 时，才显示会显示主界面。

```
MainWindow w;
Login d1;
d1.setWindowTitle("登录界面");
```

```
if(d1.exec()==QDialog::Accepted)
{
    w.show();
}
```

设置完成后，登录界面的逻辑代码就完成了。

接下来进行登录功能测试。运行售票端的 QT 程序，其效果如下图 3.5 所示。



图 3.5 登录界面效果

当输入错误的密码时，会有错误的提示性语句，如下图 3.6 所示。



图 3.6 密码输入错误

只有当用户名和密码输入正确后，才能登录进售票界面。售票界面的初始化窗口如下图所示 3.7 所示。



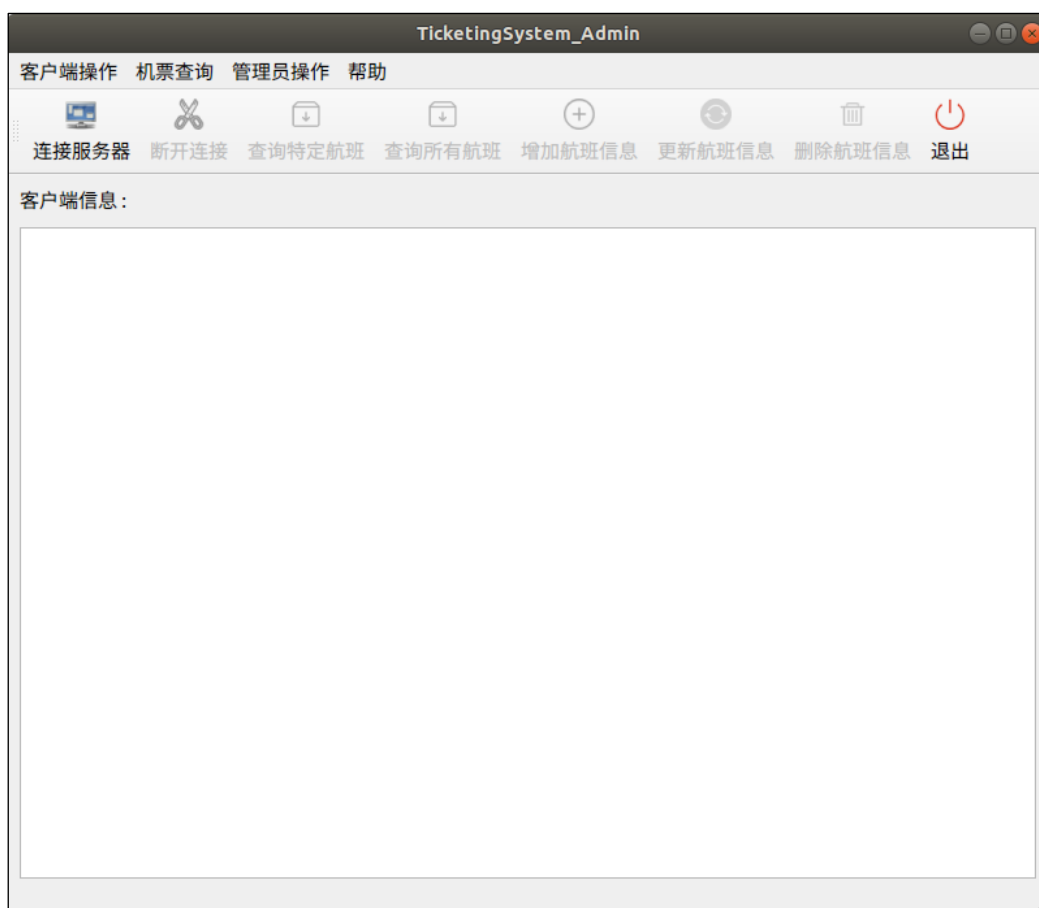


图 3.7 售票界面

### 3.2 售票界面 UI 设计

售票界面的功能函数需要引入 C 语言的代码，我们是直接将 C 语言的代码包含到了工程文件中。如下图 3.8 所示。

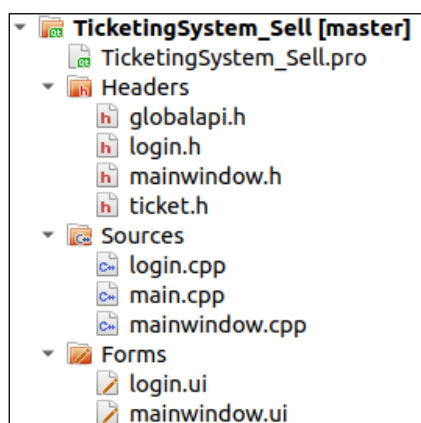


图 3.8 源代码导入

其中的 global.h 文件、ticket.h 文件是 C 语言的逻辑代码，login.h、login.cpp 和 login.ui 是登录界面的代码。因为它们与 QT 界面内容无关，所以在此部分就不再详细介绍了。

接下来介绍 UI 界面的设计过程。UI 界面的设计是在 mainwindow.ui 文件中进行

的，因为使用了 QMainwindow 类，所以可以很方便地在界面中添加菜单栏选项。首先根据售票端的功能，添加相应的菜单栏选项。售票端主要是有四个大的功能分类，分别为“客户端操作”、“机票查询”、“管理员操作”、“帮助”这四项。具体的功能见下表。

表 3.1 菜单功能表

菜单项	菜单列表	功能说明
客户端操作	连接服务器	售票端连接服务器
	断开连接	售票端断开与服务器的连接
	购买机票	售票端购买机票
	退出程序	退出售票端程序
机票查询	查询特定航班	查询指定的某个航班信息
	查询所有航班	查询所有航班信息
管理员操作	增加航班信息	增加某个航班的信息
	更新航班信息	更新某个航班的信息
	删除航班信息	删除某个航班的信息
帮助菜单	显示操作提示	显示功能说明窗口
	团队信息	显示团队信息窗口

根据上表的功能分类，设计 UI 界面的菜单布局，如下图 3.9 所示。

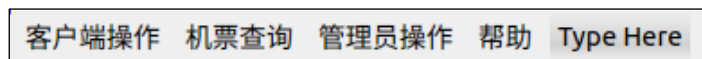


图 3.9 菜单选项

在每个菜单下，分别设置下拉菜单选项，如下图 3.10 至 3.13 所示。

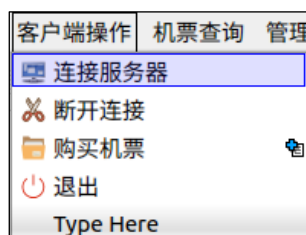


图 3.10 客户端操作菜单

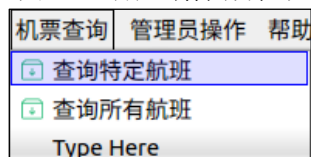


图 3.11 机票查询菜单

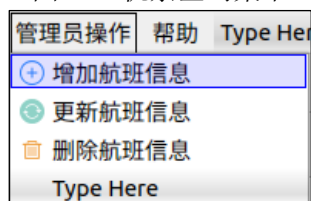


图 3.12 管理员操作菜单

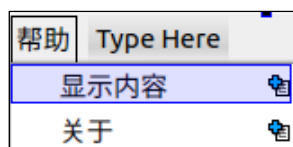


图 3.13 帮助菜单

下拉菜单设计完成后，可以为其添加图片资源。在工程文件中添加 QT 资源类，如下图所示 3.14 所示。

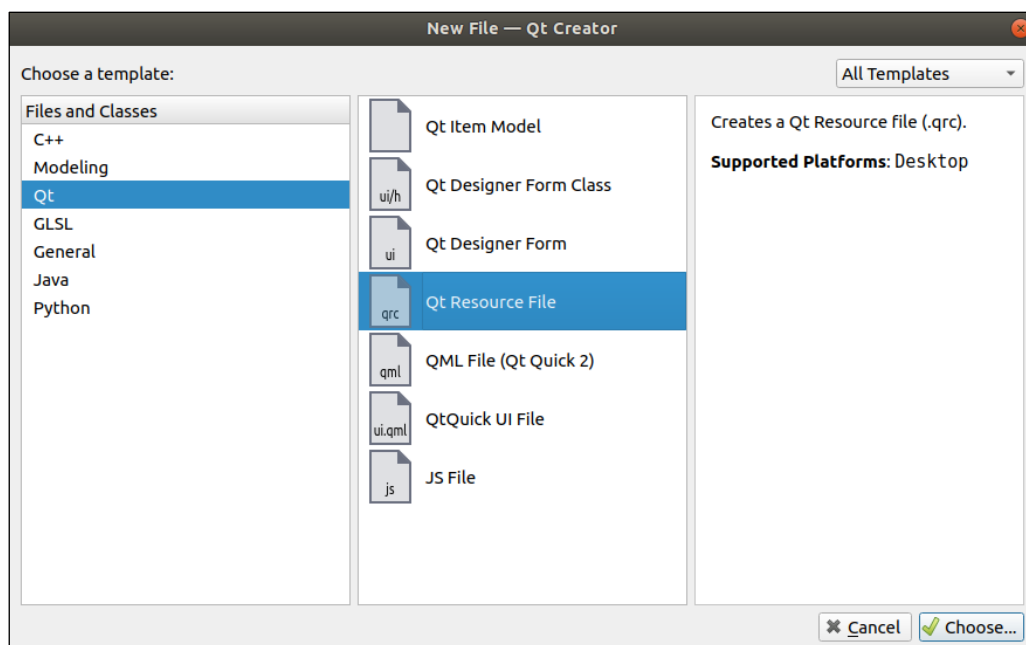


图 3.14 添加 QT 资源

添加完成的图片资源如下图所示 3.15 所示。



图 3.15 添加图片资源

首先需要在网上下载相关的图片资源，我是在阿里巴巴矢量图库里面找到的图片资源，将其下载并拷贝到 VMware 虚拟机中。如下图所示 3.16 所示。

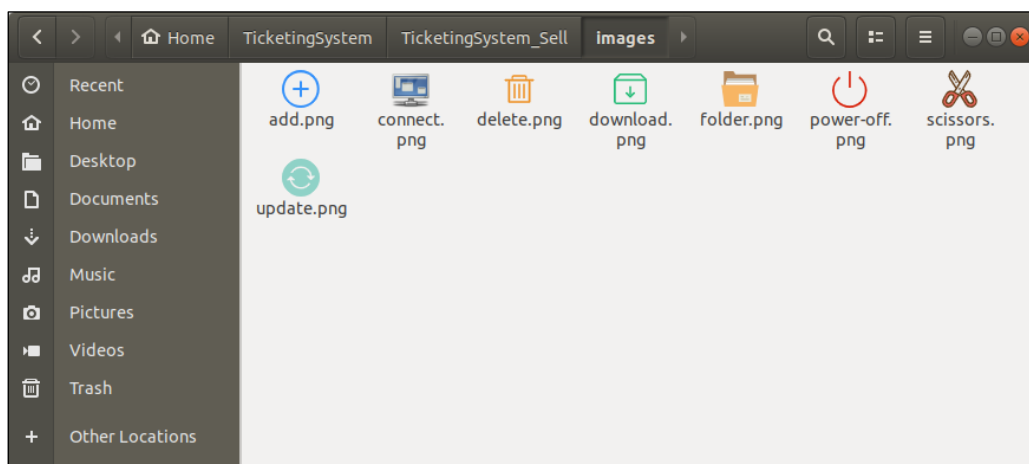


图 3.16 图片资源

接下来在编辑面板中添加图片资源段前缀和图片文件，如下图 3.17 所示。

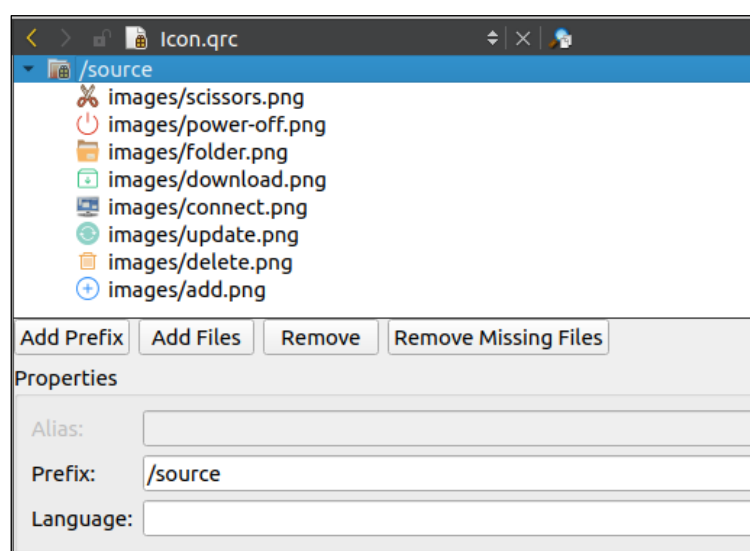


图 3.17 图片导入至工程

资源添加完毕后，工程代码中的结构如下图 3.18 所示。

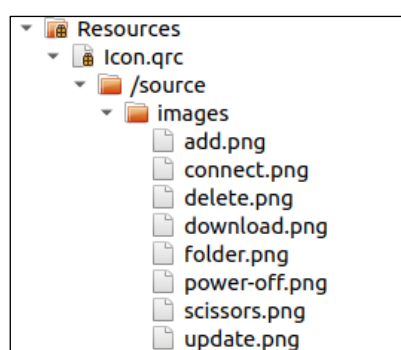


图 3.18 工程结构

接下来可以为每个下拉菜单的选项添加图片资源。在 UI 编辑界面的 Action Editor 窗口中，可以设置菜单选项的属性，如下图 3.19 所示。

Name	Used	Text	Shortcut	Checkable	ToolTip
acti...nect	<input checked="" type="checkbox"/>	连接服务器		<input type="checkbox"/>	连接服务器
acti...nect	<input checked="" type="checkbox"/>	断开连接		<input type="checkbox"/>	断开连接
acti...exit	<input checked="" type="checkbox"/>	退出		<input type="checkbox"/>	退出
act...one	<input checked="" type="checkbox"/>	查询特定航班		<input type="checkbox"/>	查询特定航班
acti...eall	<input checked="" type="checkbox"/>	查询所有航班		<input type="checkbox"/>	查询所有航班
act...how	<input checked="" type="checkbox"/>	显示内容		<input type="checkbox"/>	显示内容
acti...bout	<input checked="" type="checkbox"/>	关于		<input type="checkbox"/>	关于
act...add	<input checked="" type="checkbox"/>	增加航班信息		<input type="checkbox"/>	增加航班信息
acti...date	<input checked="" type="checkbox"/>	更新航班信息		<input type="checkbox"/>	更新航班信息
acti...lete	<input checked="" type="checkbox"/>	删除航班信息		<input type="checkbox"/>	删除航班信息
acti...cket	<input checked="" type="checkbox"/>	购买机票		<input type="checkbox"/>	购买机票

图 3.19 Action Editor 窗口

以“连接服务器”选项为例，右键单击“连接服务器”选项，选择“Edit”设置其属性。点击 Icon 选项，设置其图片文件。

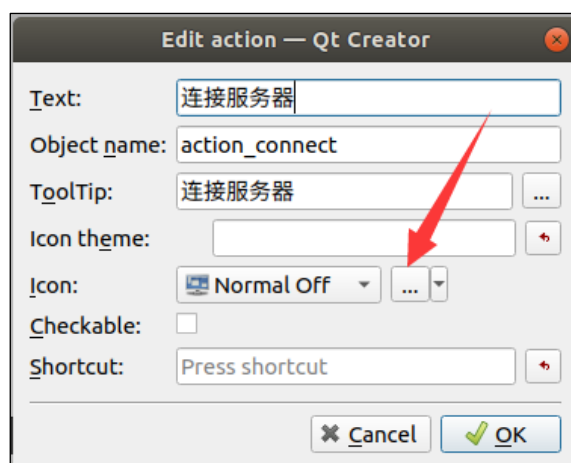


图 3.20 设置动作

在弹出的资源菜单中，选择相应的图片文件并点击 OK 即可。

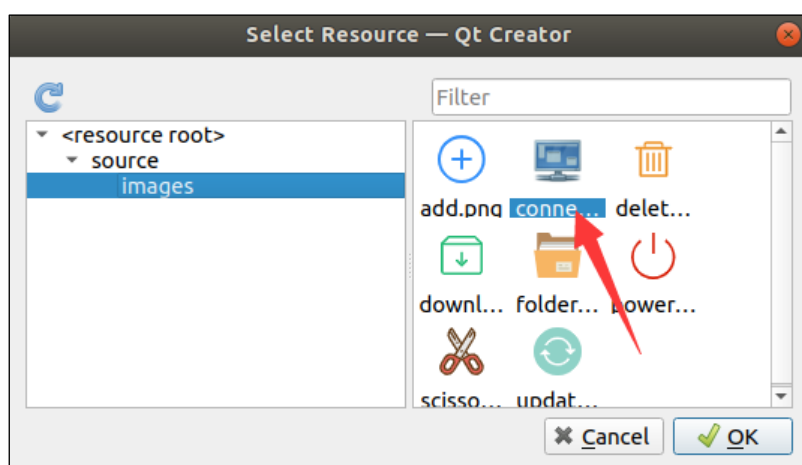


图 3.21 选择图片资源

按照这样的流程，设置所有选项的图片样式，最终的效果如下图 3.22 所示。

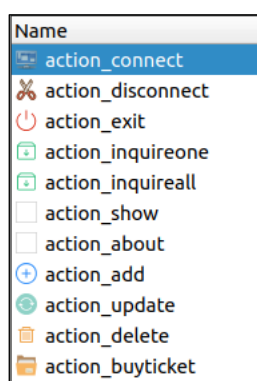


图 3.22 图片样式

其中的 action\_show 和 action\_about 选项，因为没有展示到工具栏上，所以暂时没有设置图片资源。

设置完图片资源后，可以将常用到的选项添加到工具栏中，便于用户操作。直接将 Action Editor 窗口中的常用部件拖动到工具栏中即可。因为大部分菜单选项已经设置了图片图标，所以当把选项拖动到工具栏后，相应的选项就变成了图标的形式，效果如下

图 3.23 所示。



图 3.23 工具栏效果

如果直接这样显示界面的工具栏，因为没有文字提示，可能用户不清楚每个工具选项的具体功能，所以需要为其添加文字说明。在构造函数中添加代码，设置工具栏的图片样式，如下代码所示。

```
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);
}
```

最后一行设置了工具栏的图表样式，将样式设为 `Qt::ToolButtonTextUnderIcon` 属性，其效果就是在图标的下方添加相应的文字提示。

运行代码测试一下目前的效果，首先在登录界面输入正确的用户名和密码，弹出的售票界面如下图 3.24 所示。

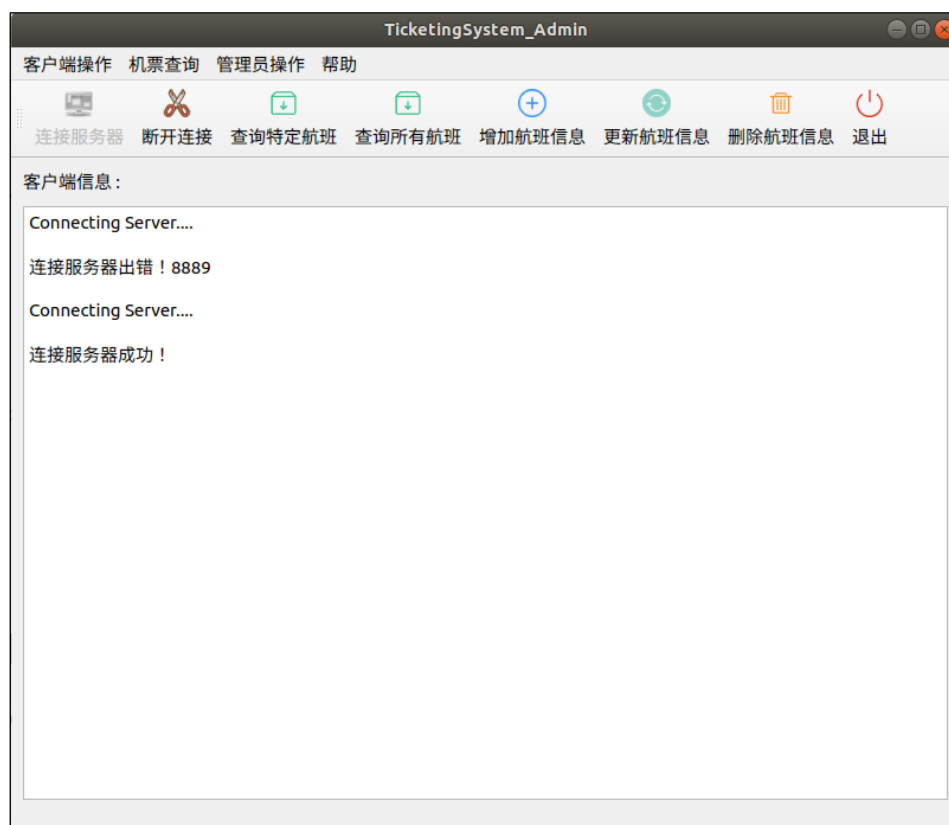


图 3.24 售票界面

### 3.3 工具栏按钮使能

可以注意到，工具栏图标文件下方已经有了相应的文字说明。另外，连接服务器后，“连接服务器”按钮就变成了灰色。接下来介绍与此相关的按钮使能操作。

售票端拥有很多功能，但是这些功能都必须要与服务器进行通信才可以实现。也就是说，当售票端没有连接服务器时，很多功能是不可用的。为了明显地提示用户，说明该功能不可用，可以将按钮使能关闭，使按钮暂时不可用。为实现此功能，可以编写一个按钮使能函数，控制按钮的使能。具体代码如下代码所示。

```
// 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);
```

上述代码中，使用布尔类型的变量 `boolean` 配合 `setEnabled` 方法，实现了按钮的使能控制。需要注意的是，“退出”按钮应该是一直保持使能状态，在任意时刻都是可用的，所以在代码中没有涉及“退出”按钮。“退出”按钮保持默认使能状态即可。

如果要改变按钮的使能状态，无非就只有两种情况，也就是在“连接”按钮和“断开连接”按钮被按下的时刻。因为只有这两个按钮可以改变售票端与服务器之间的连接状态。我们可以通过设置一个全局变量 `isconnected`，在任意时刻获取与服务器之间的连接状态。

首先在启动售票端伊始，可以先将其余按钮失能，只让“连接”按钮使能。第二是在“连接”按钮点击并成功连接服务器后，让“连接”按钮失能，其余按钮使能。第三就是在“断开连接”按钮被按下后，让“连接”按钮使能，其余按钮失能。通过调用 `enable_button` 函数，就可以很方便地实现该功能。

下面进行按钮使能功能的测试。

当刚刚开启售票端时，按钮的使能状态如下图 3.25 所示。

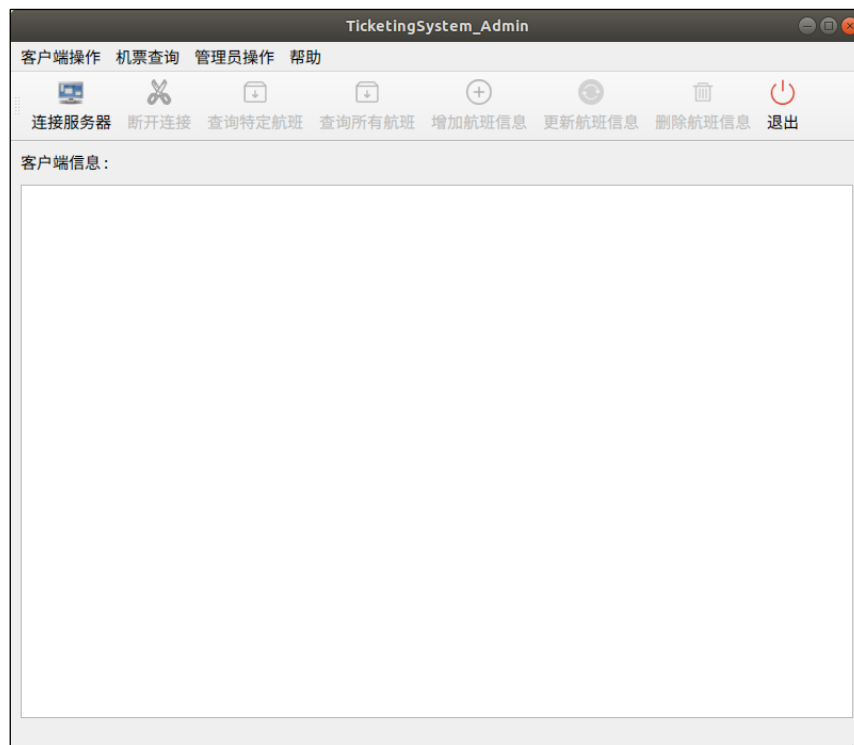


图 3.25 按键失能效果

点击“连接服务器”按钮后，按钮使能状态如下图 3.26 所示。

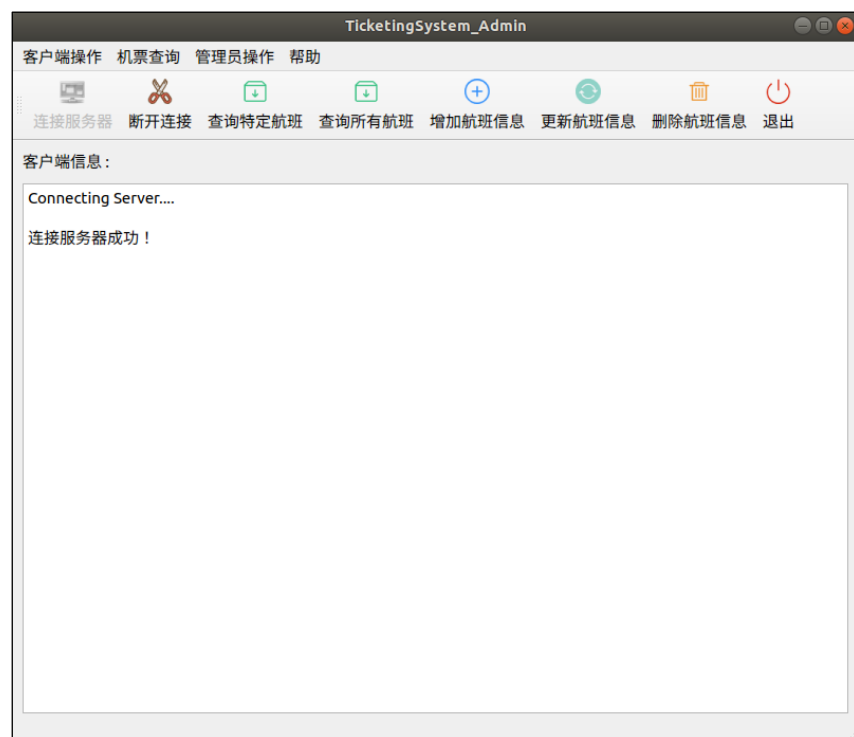


图 3.26 按键使能效果

可以看到，按钮的使能功能是正常的。

至此，QT 界面的基本功能就实现了，包括添加工具栏、设置工具栏图表样式、设置按钮使能功能等等。接下来需要实现具体的通信功能逻辑，实现售票端与服务端之间的通信过程。



### 3.4 数据输出函数

在介绍具体的槽函数编写之前，需要考虑文本显示的问题。C 语言底层的逻辑代码输出信息时使用了 `printf` 语句，将文本信息直接输出到了标准输出流，也就是终端显示屏中。而 QT 中的输出是要输出到文本框组件中，这就需要进行数据的输出转换。编写相应的函数，如下代码所示。

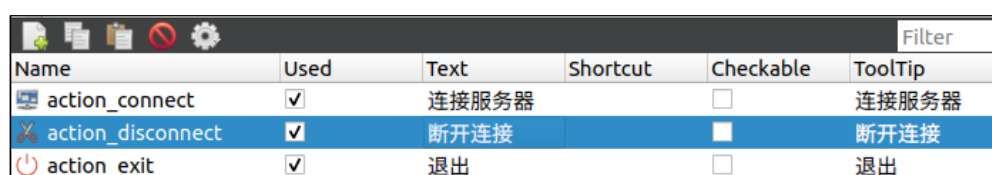
```
void MainWindow::display_info(QString msg){
    ui->textBrowser->append(msg);
}
```

该函数比较简单，是直接调用 `append` 方法，将形参里的 `msg` 变量内容追加到了 `textBrowser` 文本框中。以后每当底层逻辑代码要输出信息时，直接调用该函数即可实现信息输出。

### 3.5 函数命名空间

通信逻辑代码的实现，也无非就是对 QT 界面按钮的槽函数进行编写。因为我们已经有了 Linux C 的底层通信逻辑代码，所以实现底层通信只需要将原有的 C 代码移植到 QT 对应的槽函数中即可。我们需要为所有的按钮都设置槽函数，有些按钮的槽函数内容涉及 C 语言的底层逻辑，其中的某些函数可能会与 QT 中的函数冲突，下面以“断开连接”按钮的槽函数为例，介绍函数冲突时的解决方法。其余的涉及 C 语言的内容就不再赘述了。

首先在 UI 编辑界面，找到 Action Editor 窗口，在该窗口中右键单击“断开连接”选项，选择“Go to slot”转到它的槽函数。



Name	Used	Text	Shortcut	Checkable	ToolTip
action_connect	<input checked="" type="checkbox"/>	连接服务器		<input type="checkbox"/>	连接服务器
action_disconnect	<input checked="" type="checkbox"/>	断开连接		<input type="checkbox"/>	断开连接
action_exit	<input checked="" type="checkbox"/>	退出		<input type="checkbox"/>	退出

图 3.27 转到槽函数

```
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);
```

如上代码所示，在槽函数中要实现断开与服务器连接的逻辑功能，需要用到 `close` 函数。因为 QT 本身也有 `close` 函数，那是用来关闭当前窗口的，而我们使用的 C 语言

中的 close 函数是用来关闭与服务器之间的套接字连接的，两个函数名有冲突时，可以在 close 函数前面加上两个冒号，改变当前函数的命名空间，声明要使用 C 语言中的 close 函数。这样，函数名冲突的问题就可以被解决。

### 3.6 信息交互界面

售票端的各种按钮，包括“连接服务器”、“断开连接”、“购买机票”、“退出”、“查询特定航班”、“查询所有航班”、“增加航班信息”、“更新航班信息”和“删除航班信息”这几个按钮，它们的槽函数编写都是移植的 C 语言代码，需要注意的是，像“购买机票”、“查询航班信息”这类的功能，都需要与用户进行交互，需要用户输入具体的航班信息等内容。在 C 语言中，我们直接在终端中输入即可，但在 QT 中，需要弹出相应的对话框，提示用户输入。下面以“增加航班信息”功能为例，介绍如何与用户实现信息交互。

管理员在点击“增加航班信息”按钮后，需要在弹窗中输入具体的航班信息。为此，我们需要为其新建一个 QDialog 对话框。关键代码如下代码所示。

```
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:"));
```

这里使用了 QDialog 和 QFormLayout 的对象，分别创建了对话框和消息提示列表。使用 connect 连接信号与槽函数，当用户点击确定按钮时，获取用户在对话框中所输入的文本信息内容。

下面介绍一下“显示帮助信息”和“关于”按钮的槽函数编写过程。这两个按钮被按下后，都需要显示提示信息框，以“显示帮助信息”按钮为例，其代码如下代码所示。

```
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(&buttonBox);
```

```
QObject::connect(&buttonBox, SIGNAL(accepted()), &dialog, SLOT(accept()));
if (dialog.exec() == QDialog::Accepted) {
    display_info("查询信息成功\n");
}
```

这里使用了 QDialog 对话框来显示弹出的提示信息，使用 QFormLayout 类的对象 form，控制每条文本的格式。每条文本以 QLabel 的形式被创建，QLabel 支持 HTML 格式，所以我这里使用了<h1>标签将文本放大，<center>标签将文本居中。每一条文本使用 addRow 方法添加到 form 对象中，最后使用 buttonBox 与 form 绑定，为界面添加按钮组件。使用 connect 连接信号与槽，用 dialog.exec()的值判断用户是否点击确认按钮。用户点击确认后，在前端界面显示提示信息。

“关于”按钮也是相同的处理方式，其具体代码如下图所示。当用户点击“关于”按钮时，同样会显示提示信息框。其中包括了程序的名称、版本号以及团队信息和更新日期。用户点击确认按钮后，会在前端界面显示提示信息。

```
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");
}
```

## 4 实现的效果

代码编写完成后，运行售票端进行测试。下面仅介绍售票端实现的效果，服务端和购票端效果请查看第七章附录中的测试视频链接。

### 4.1 登录界面及初始化

启动售票端后，首先会出现管理员的登录界面，如下图 4.1 所示。



图 4.1 登录界面

如果输入错误的用户名或密码，则登录失败，售票端弹出提示信息。



图 4.2 提示信息

如果用户名和密码输入正确，则窗口跳转到售票界面。售票界面初始化时只能点击“连接服务器”和“退出”按钮。其界面如下图 4.3 所示。

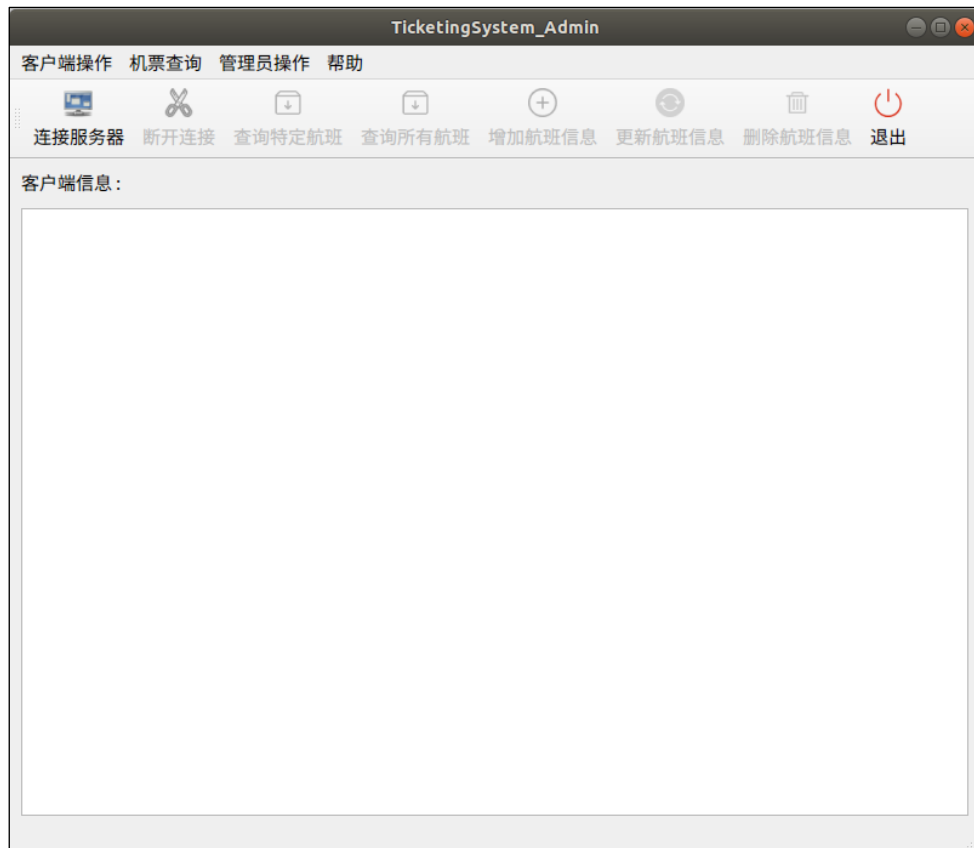


图 4.3 售票界面初始化

此时需要启动服务端程序，开启服务器才能在售票端成功连接服务器。

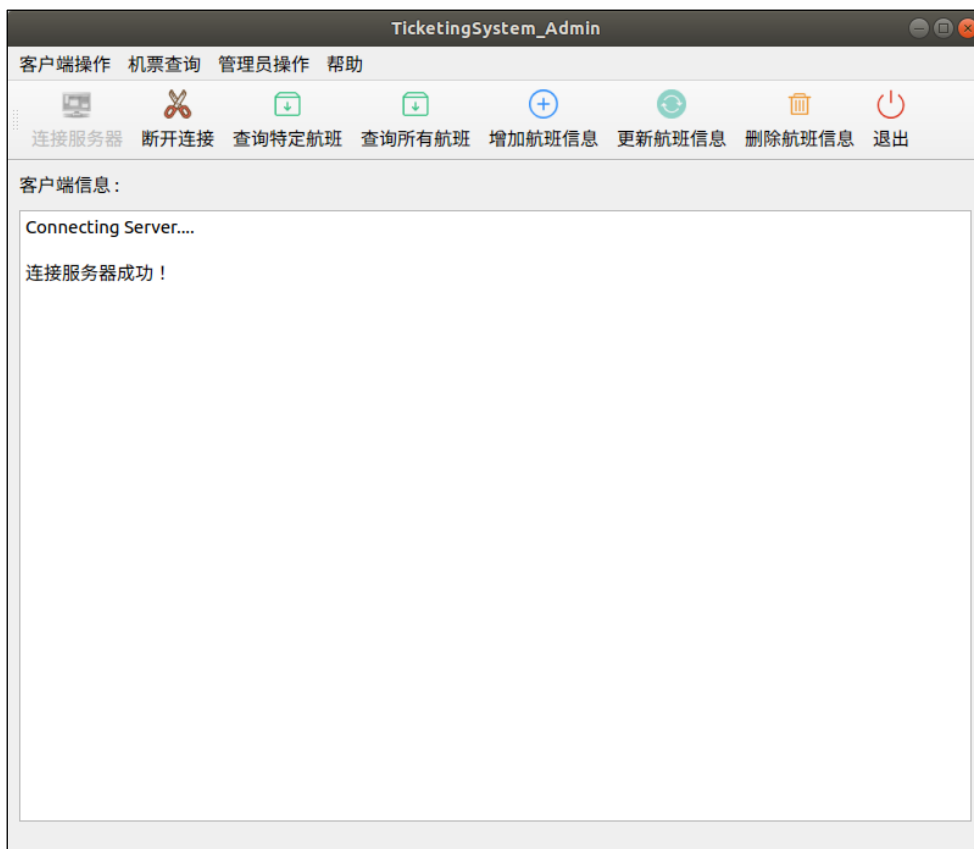


图 4.4 售票界面连接服务器

## 4.2 查询特定航班

用户点击“查询特定航班”按钮，会弹出提示窗口，要求用户输入待查寻的航班号信息，如下图 4.5 所示。用户在弹出的提示窗口中输入航班号，点击 OK 按钮即可进行航班查询。如果用户点击 Cancel 取消按钮，则取消当前的查询功能，返回到程序主界面。

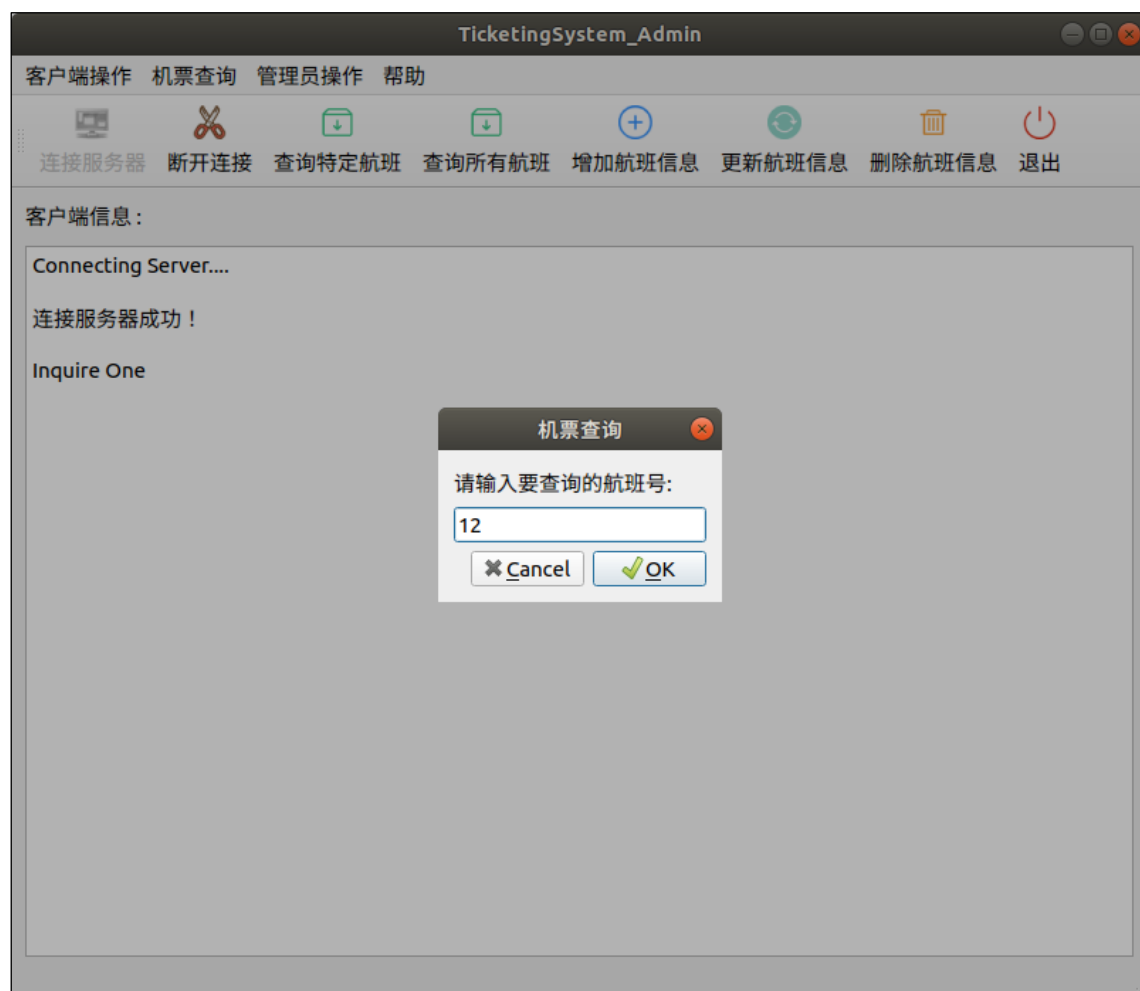


图 4.5 输入航班号

当该航班存在，即用户输入的航班信息无误时，售票端会显示航班信息。

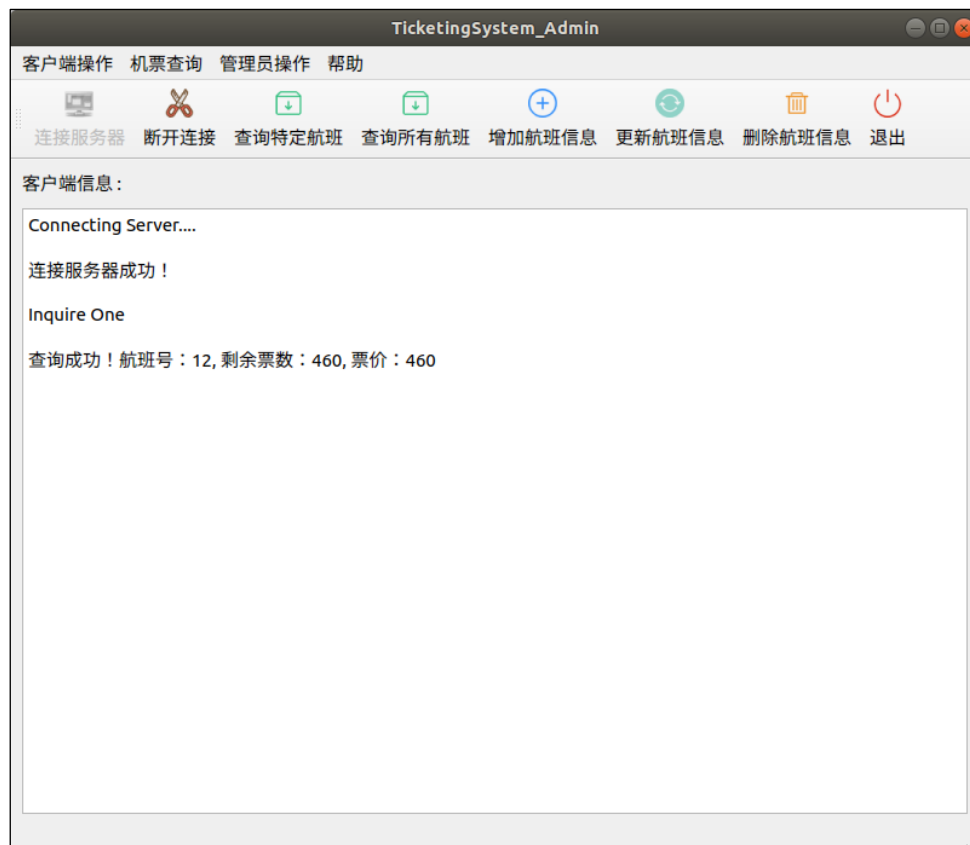


图 4.6 显示航班信息

当用户输入的航班号有误时，售票端显示警告信息。

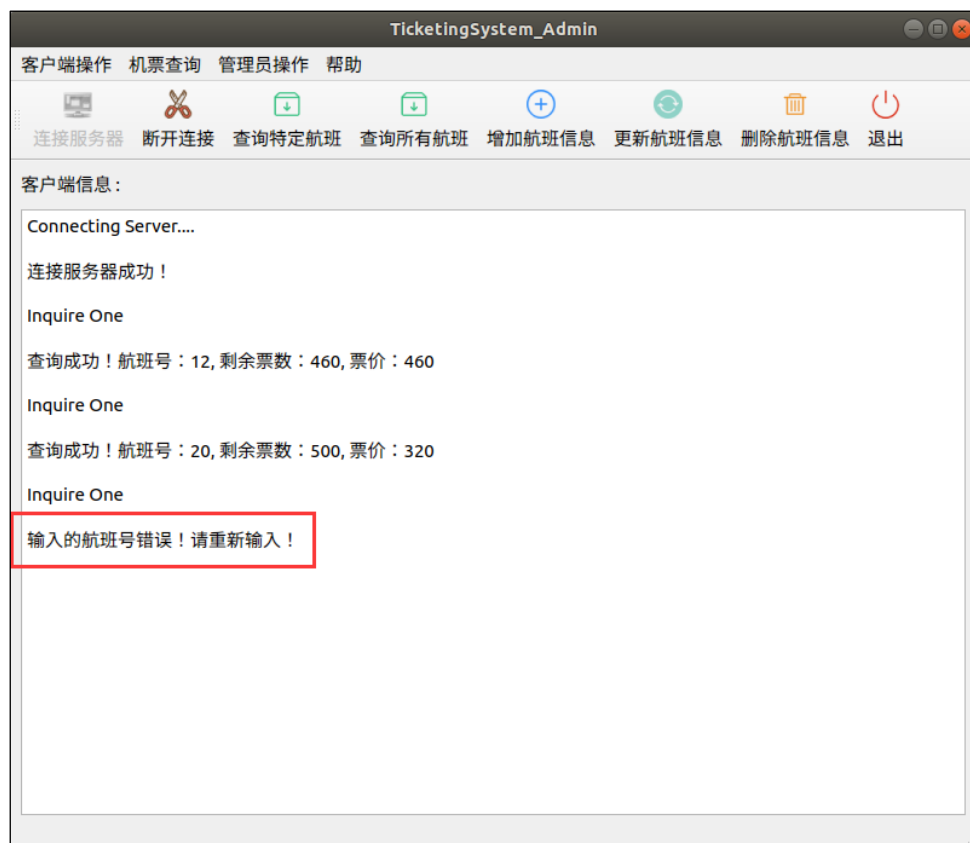


图 4.7 显示警告信息

### 4.3 查询所有航班

点击“查询所有航班”按钮可以查询所有存在的航班信息。如下图 4.8 所示。当航班信息较多时，窗口会自动显示滚动条。可以通过拖动滚动条查看所有航班的具体信息。

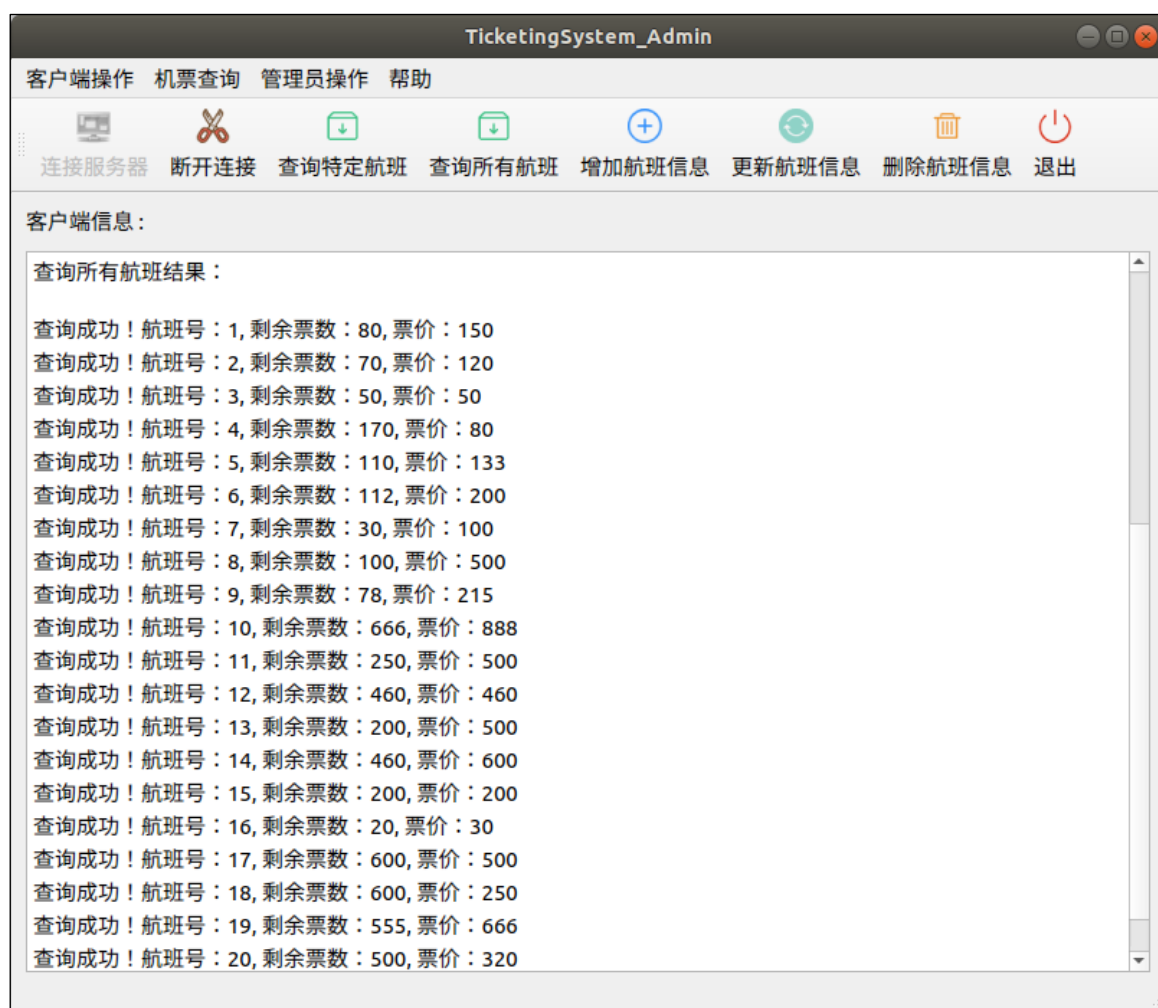


图 4.8 查询航班结果

### 4.4 增加航班信息

点击“增加航班信息”按钮，会弹出提示窗口，要求管理员输入待增加的航班信息。当管理员输入有效的航班信息时，点击 OK 即可确认提交。当管理员点击取消按钮，可以取消当前的操作。



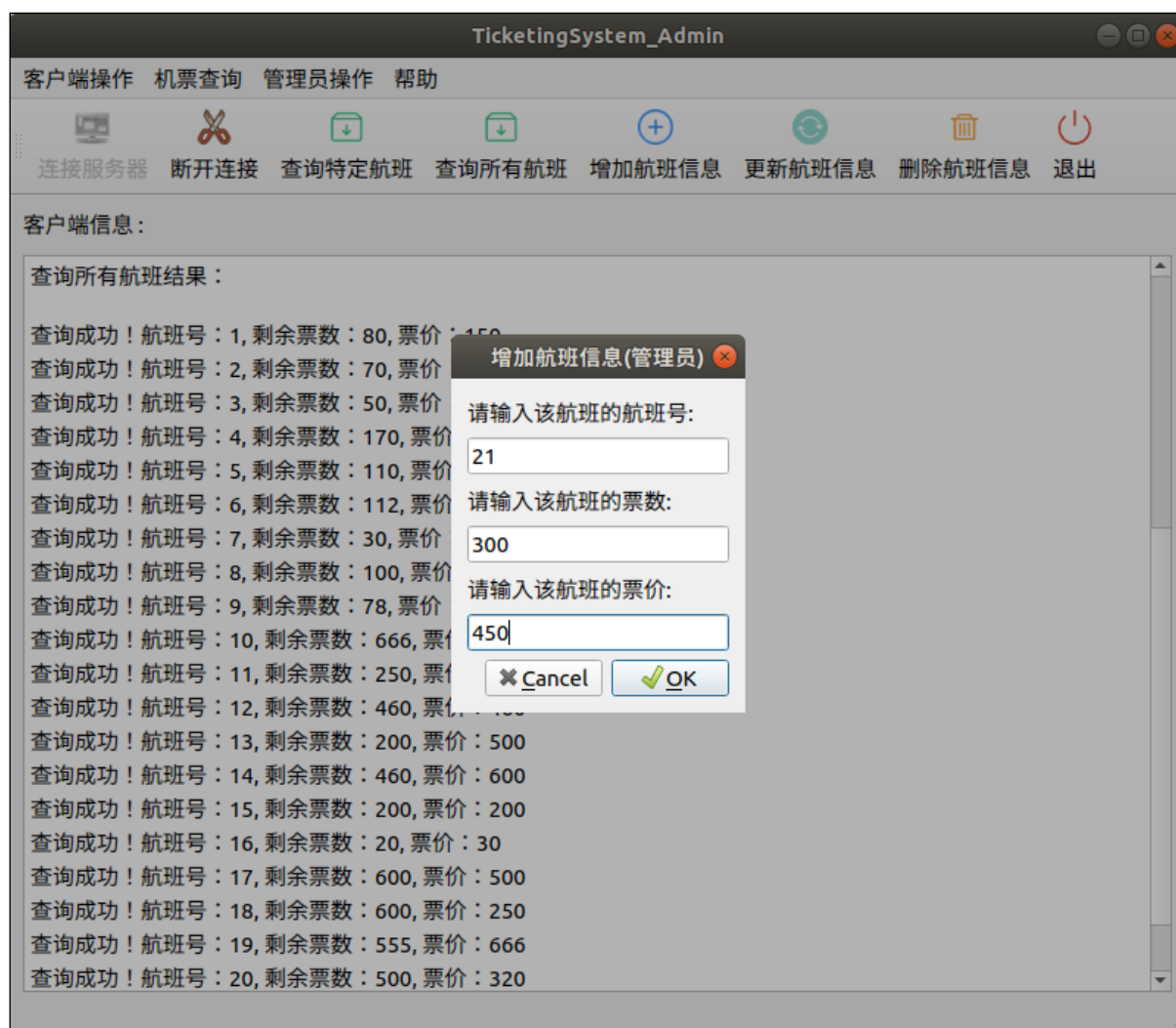


图 4.9 增加航班信息

现测试增加第 21 号航班，票数为 300，票价为 450，点击 OK 确认提交信息。此时售票端会显示增加航班信息成功的提示信息。效果如下图 4.10 所示。

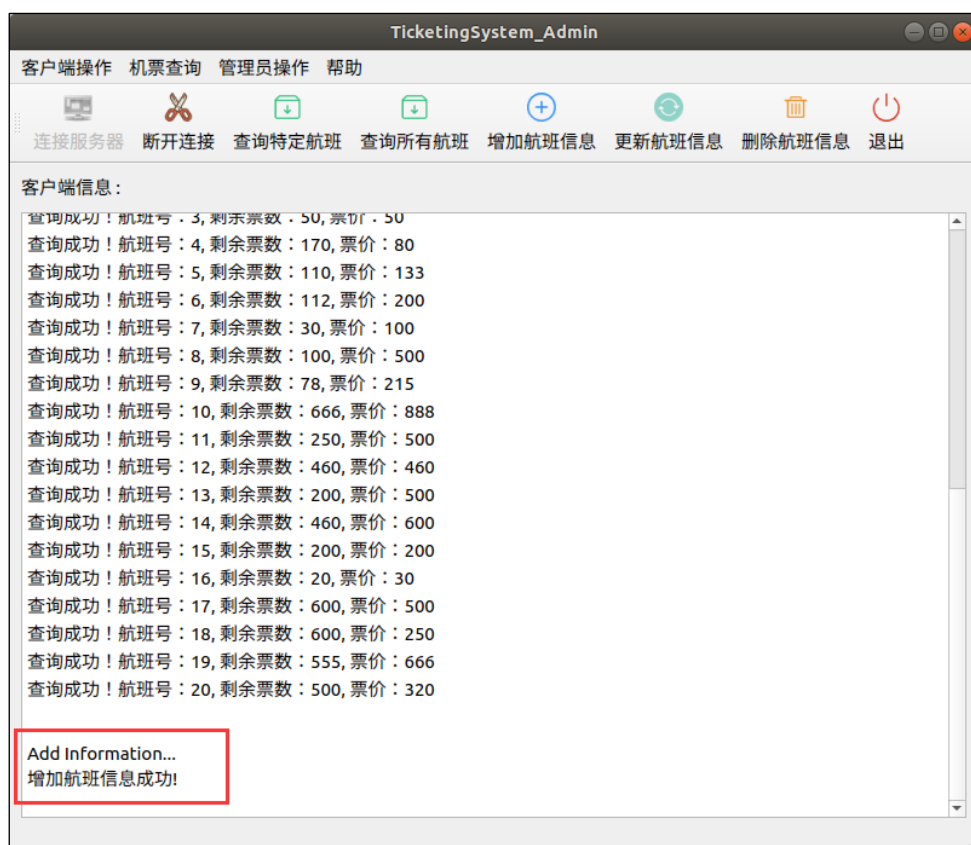


图 4.10 增加航班提示

再次查询所有航班信息，可以看到第 21 号航班的信息已经被添加进来了。

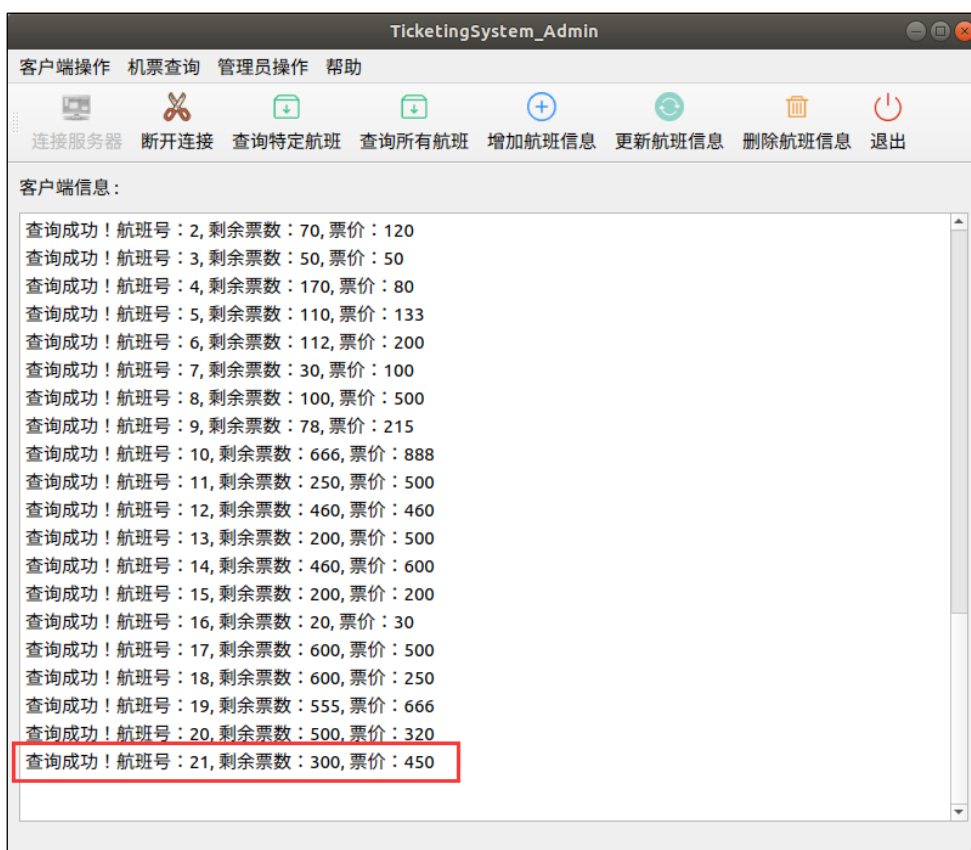


图 4.11 查询航班信息

## 4.5 更新航班信息

管理员点击“更新航班信息”按钮后，同样会弹出提示窗口，要求管理员输入待更新的航班信息，如下图 4.12 所示。当管理员输入有效的航班信息时，点击 OK 即可确认提交。当管理员点击取消按钮，可以取消当前的操作。

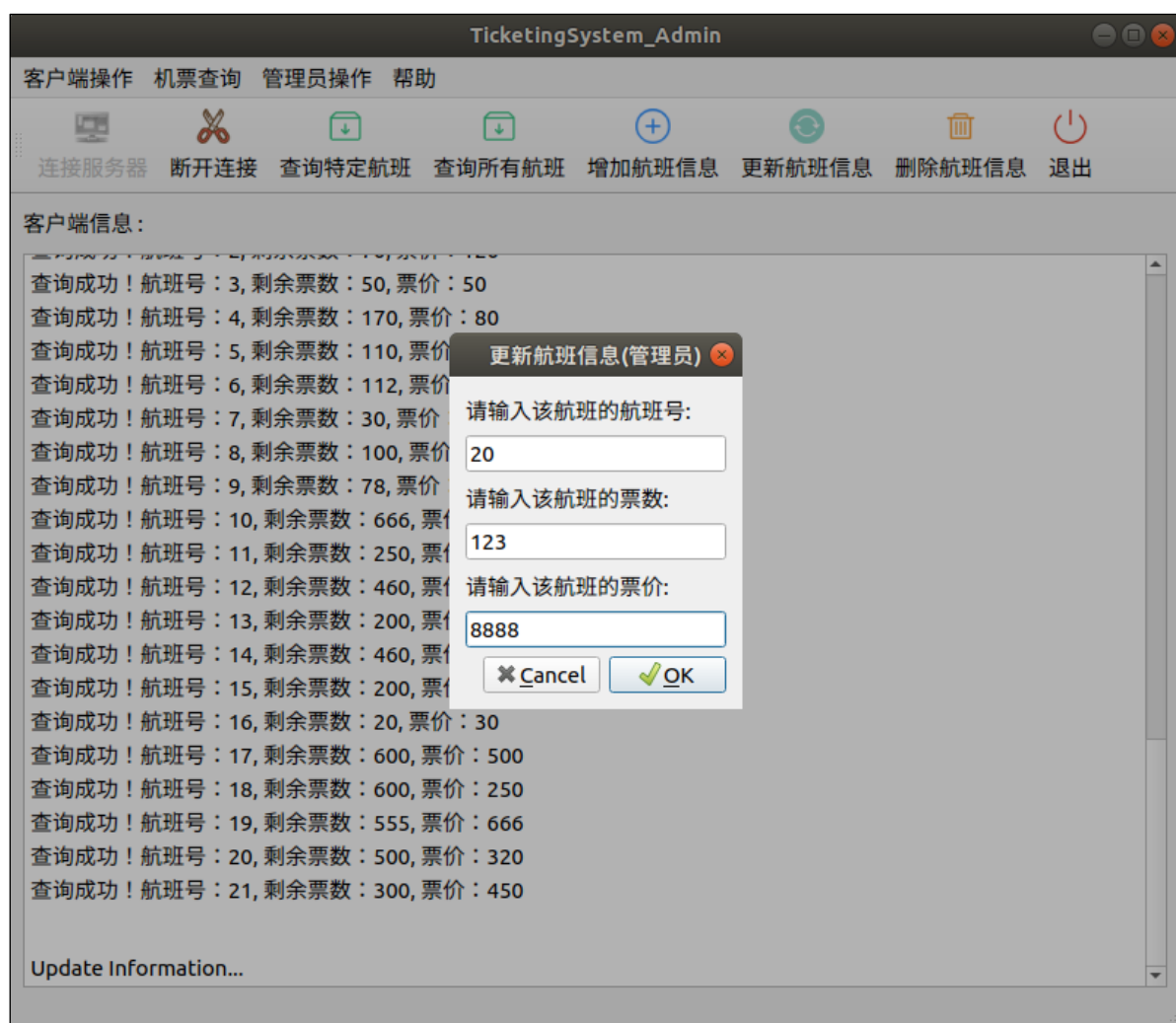


图 4.12 更新航班信息

测试更新一下第 20 号航班，将票数改为 123，票价改为 8888，点击 OK 提交。再次查询所有航班信息，可以看到第 20 号航班的信息已经被更新，如下图 4.13 所示。

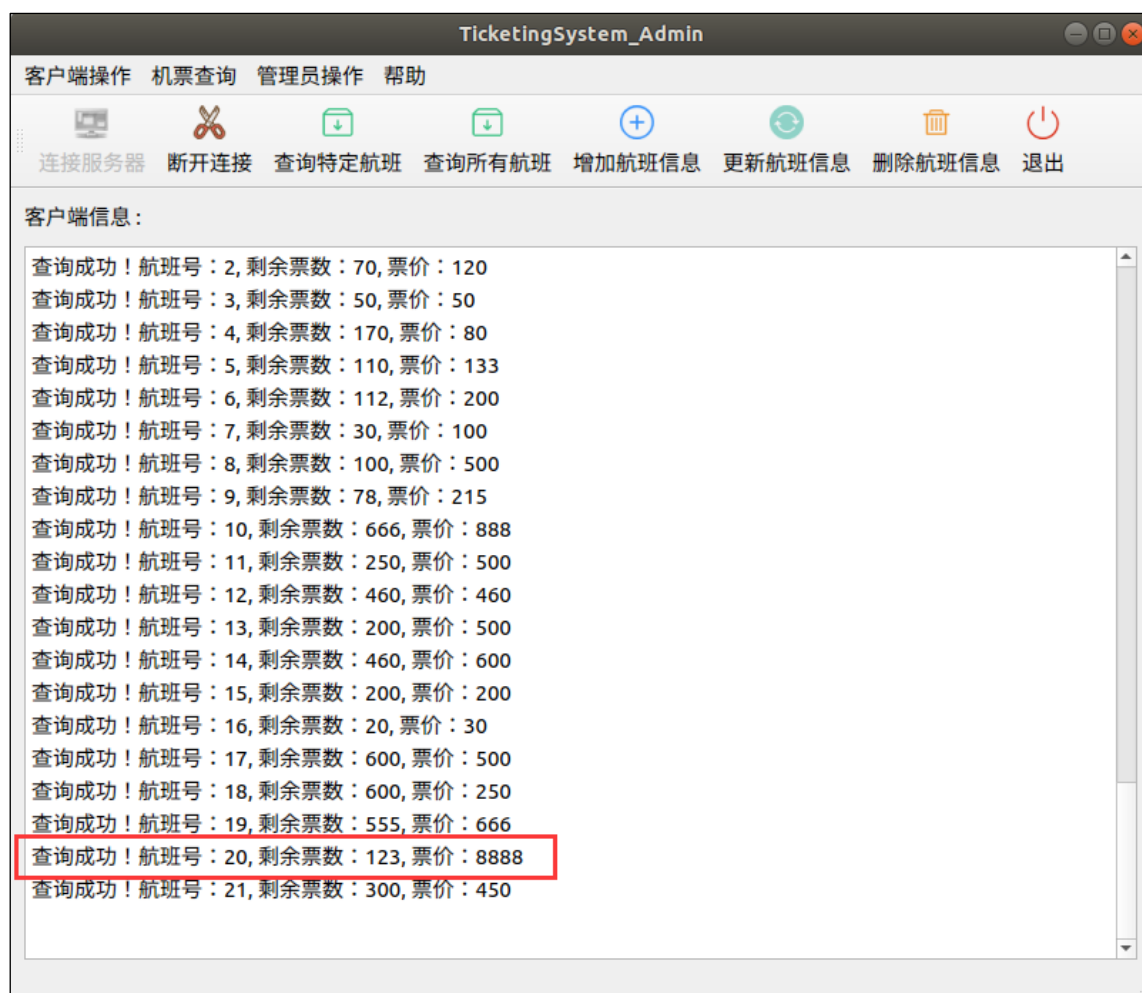


图 4.13 查询航班信息

#### 4.6 删除航班信息

管理员点击“删除航班信息”按钮，会弹出提示框，要求管理员填写待删除的航班号。如下图 4.14 所示。当管理员输入有效的航班号时，点击 OK 即可确认提交，删除对应的航班信息。当管理员点击取消按钮，可以取消当前的操作。

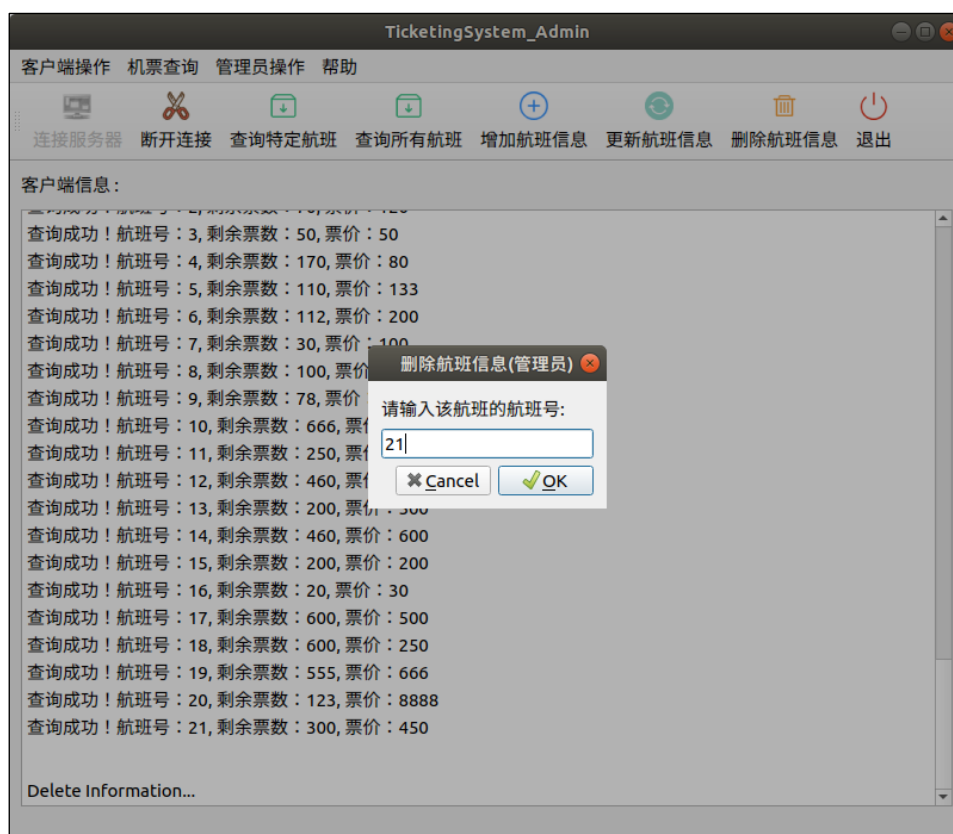


图 4.14 删除航班信息

我们测试输入第 21 号航班，再次查询所有航班，航班信息如下图 4.15 所示。

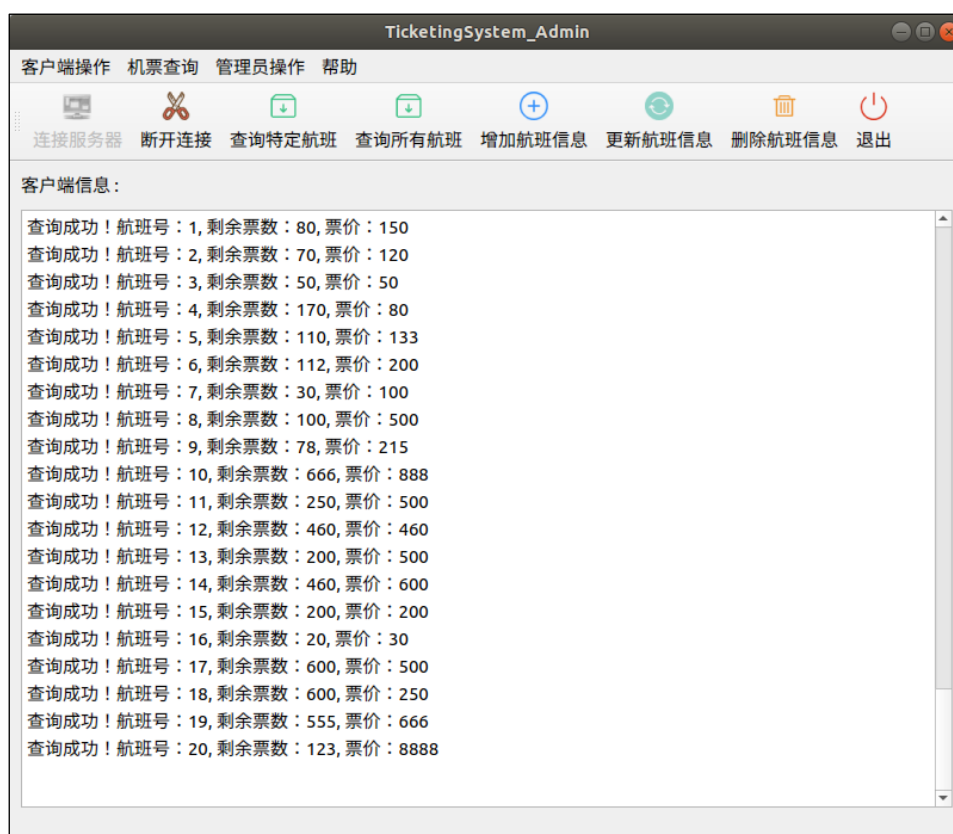


图 4.15 查询航班信息

可以看到，现在第 21 号航班的信息已经被删除了。

## 4.7 帮助信息和退出程序

点击菜单栏的帮助信息按钮，在下拉菜单中找到“显示内容”选项，可以看到程序功能说明，如下图 4.16 所示。

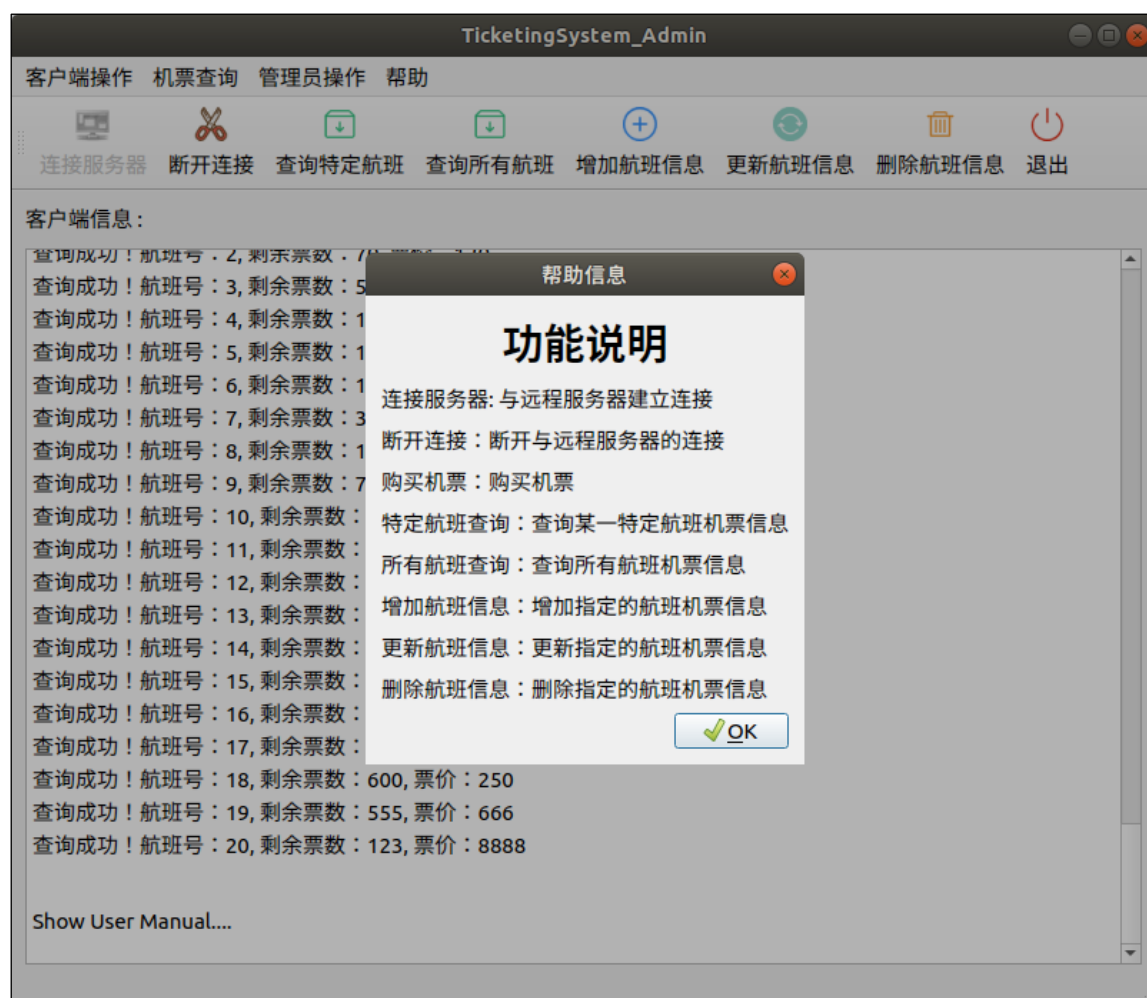


图 4.16 显示功能说明

在“帮助”的下拉菜单中，找到“关于”选项，可以找到程序的版本信息和我们团队的信息，如下图 4.17 所示。



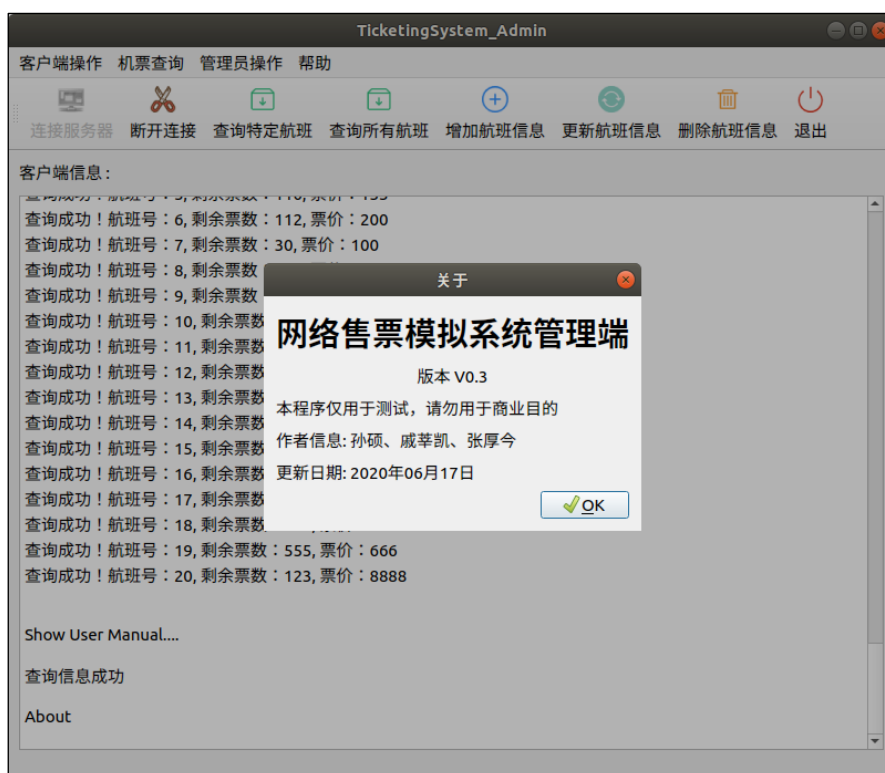


图 4.17 显示团队信息

点击“断开连接”按钮，可以断开售票端与服务器直接的连接。此时售票端界面显示提示信息，同时按钮使能发挥作用，只有“连接服务器”和“退出”按钮使能，其余按钮变为失能状态。

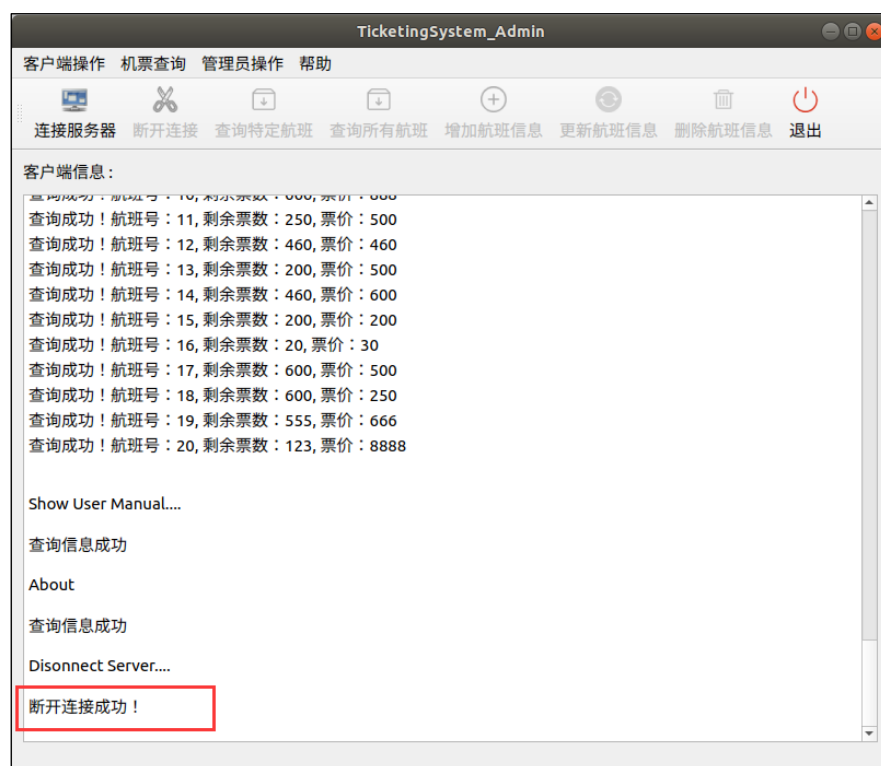
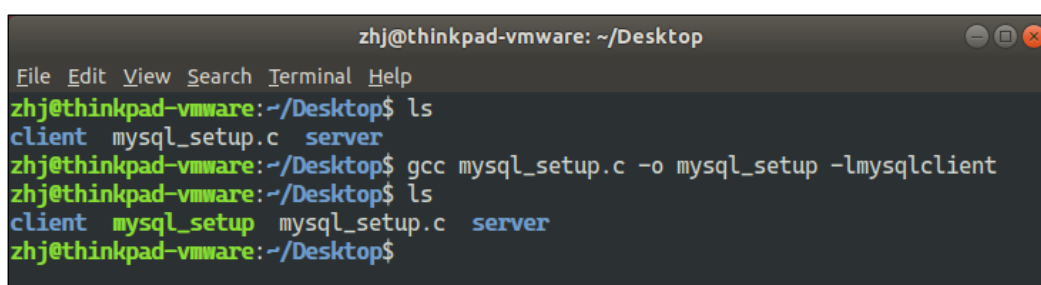


图 4.18 断开连接

点击工具栏最右侧的“退出”按钮，可以退出当前窗口。

## 5 实训中遇到的问题及解决方案

实训中遇到了很多问题，大部分都是细枝末节的小问题。通过团队之间的讨论，都已经被解决了。其中我印象较深的是有关数据库的一个问题。我们团队制作的航班售票系统，其航班数据是被存储在了 MySQL 数据库中。在纯 Linux C 的环境下编译代码时，直接使用 MySQL 的链接库编译代码即可，如下图 5.1 所示。

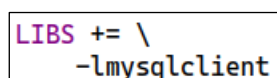


```

zhj@thinkpad-vmware: ~/Desktop
File Edit View Search Terminal Help
zhj@thinkpad-vmware:~/Desktop$ ls
client mysql_setup.c server
zhj@thinkpad-vmware:~/Desktop$ gcc mysql_setup.c -o mysql_setup -lmysqlclient
zhj@thinkpad-vmware:~/Desktop$ ls
client mysql_setup mysql_setup.c server
zhj@thinkpad-vmware:~/Desktop$
    
```

图 5.1 终端代码编译

但是将 Linux 代码移植到 QT 环境中之后，编译 QT 工程时总是报错，说是 MySQL 数据库的头文件<mysql/mysql.h>不存在。后来经过查阅资料和团队讨论才明白，需要在 QT 的配置文件添加上 MySQL 的链接库才行，如下图 5.2 所示。

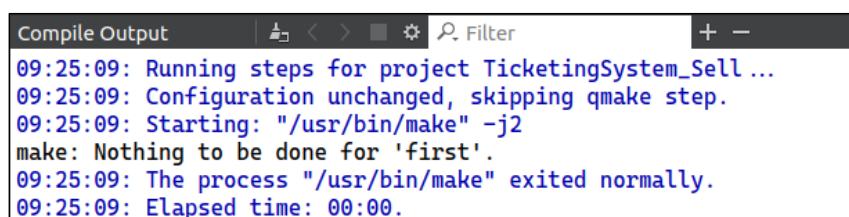


```

LIBS += \
        -lmysqlclient
    
```

图 5.2 添加 QT 依赖库

添加上链接库后，再编译代码就不会报错了。在 QT for Linux 中编译程序，就没有问题了，如下图 5.3 所示。



```

Compile Output
09:25:09: Running steps for project TicketingSystem_Sell...
09:25:09: Configuration unchanged, skipping qmake step.
09:25:09: Starting: "/usr/bin/make" -j2
make: Nothing to be done for 'first'.
09:25:09: The process "/usr/bin/make" exited normally.
09:25:09: Elapsed time: 00:00.
    
```

图 5.3 编译成功



## 6 实训总结

本次实训制作了一个简易的航班购票模拟系统，使用 Linux C 编写后台程序，由 QT for Linux 软件编写前端界面，实现了后台功能与前端界面的完美结合。在 QT 编程方面，由我负责制作的售票端模块涉及到了登录界面设计、图片资源引入、菜单及工具栏管理、按钮使能、函数命名空间转换、交互界面设计等多方面的工作，涉及到很多日常学习中经常遇到的知识点。

通过本次实训学习，不仅使我对 QT 的基础操作有了更深入的理解，也为我在编写图形界面的技术思路积累了宝贵的经验。日后的项目开发必定离不开图形界面编程，本次实训使我真正感受到了图形界面编程的强大。

在团队合作方面，本次实训也使我受益匪浅。我们团队三人分工合作，项目进行地有条不紊，整个项目实施过程也非常顺利。实训前期时我们的项目整体进展比较缓慢，后期加快了项目进度，在规定时间内顺利完成了实训。遇到问题时我们会及时沟通、积极讨论，从而产生了很多有价值的新思路和新创意，像“客户端的欢迎页面”、“售票端的注册页面”、“航班信息采用数据库存储”等等技术思路，我觉得这些都是非常有价值的团队成果。感谢团队成员孙硕和戚莘凯为项目做出的宝贵贡献，相信在以后的学习和生活中，本次实训一定会对我们产生积极的影响。