目录

目录

基于 Qt 的跨平台多媒体播放器

摘要本文旨在通过 Qt Creator 开发平台开发出一款简单易用的多媒体播放器。该播放器基本功能包括了：音频和视频文件的打开、播放本地音频和视频文件、播放服务器上的音频和视频文件、拖动进度条改变播放时间等。

该播放器使用 windous 7 系统作为客户端开发平台，Ubuntu 系统作为服务器平台。编程语言使用的是 Visual C++ 6.0 通过 Qt 平台进行开发。拥有友好的客户端界面和稳定的运行能力。

关键词：Qt Creator 开发平台；多媒体播放器

Cross-platform multimedia player based on Qt.

Abstract: This paper aims to develop a simple and easy-to-use multimedia player through the Qt Creator development platform. Basic functions include: the player, audio and video files open, played the local audio and video files, play audio and video files on the server, drag the progress bar change playing time, etc.

The player USES the windous 7 system as the client development platform and the Ubuntu system as the server platform. The programming language USES Visual C++ 6.0 to be developed through the Qt platform. Have a friendly client interface and stable operational capability.

Key words: Qt Creator development platform; multimedia player

第一章 序言

1.1 引言

随着计算机大量的出现和广泛地应用使用以及互联网技术的不断进步，人们的生活节奏和工作效率都得到了很大的进步。在个人计算机越来越普及的今天各种各样的软件不仅仅提高了人们的工作效率，同时也极大程度的丰富了人们文娱生活。人们的生活品质不断得到提高，所以对业余生活的品质要求也越来越高，而看视频追剧，追番正是人们众多娱乐项目中的一种。同时因为自媒体的出现也大大提高了人们对视频类的媒体文件接触得机率。因而各种各样的播放器也随之出现

但是目前市面上的播放器因为一味的追求功能的强大和外观的华丽，反而会给用户的使用带来许多不便。比如说一些播放器在播放网络资源的时候会启动网络加速器，这对于其他用户来说是十分不利的，或造成网络的拥堵。虽然现在的网速都普遍很快，但这也是一种资源的浪费。再比如说一些平时用户根本使用不到的功能会跟随播放器的启用而启动，这对于他们的计算机来说是一种严重的资源占用，非常不可取。

1.2 背景与意义

信息时代的飞速发展使得人们能够通过越来越多的方法来获取信息，传统的信息传递方式是通过文字的记录来实现的，这种方式已经被人们沿用上千年。随着计算机的出现和飞速的发展越来越多的信息传递方法被人们广泛的使用着。而多媒体技术正是由此而诞生，多媒体不仅仅有文字还有其他的方式例如视频、音频、图片等等。能够处理这些文字、视频、音频、图像的计算机就叫做多媒体计算机。

这种计算机有一项很重要的功能就是对多媒体文件的播放，而视频、音频正是其中主要格式。目前市面上就充斥着各式各样的播放器，这些播放器拥有者华丽的界面，强大的功能。但与此同时强大功能和华丽外观带来的真是是计算机资源的一种占用。

Qt Creator 开发的播放器不仅小巧而且可与方便移植到不论是 Windows 还是 Linux 各种平台上。同时他也能很好的移植到嵌入式系统中。因此开发 Qt 程序有着非常重要的意义，也有着非常大的实用价值。

1.3 市面上常见的播放器调查

（1）QQ 影音

QQ 影音是腾讯公司旗下推出的本地播放器能够支持任何格式的影片视频文件。首创了轻量级多播放内核技术，发挥了显卡的硬件加速的功能。

（2）PPTV

PPTV 又叫 PPLive 是上海聚力传媒公司开发的在线视频软件，它的特色是视频直播。并且通过网页、客户端、App、机顶盒等渠道为用户提供网络视频播放服务。

（3）PPS

PPS 是一个集合了 P2P 点播直播于一体的网络电视软件，播放流畅，由于采用了 P2P 的技术所以也多人观看越流畅。

1.4 国内外研究现状

计算机技术的飞快发展和应用，让个人计算机飞快的在普通人家普及开来。随着越来越多的人接触互联网，网络的规模也在逐渐扩大，越来越多的资源也出现在网上。大量的计算机应用软件也随之被人们开发使用。在这一趋势的下娱乐软件业发展出了一片广阔的天地。这些软件极大程度的丰富了人们的生活，提高了人们的生活质量。

现在市面上拥有着大量的各具特色的播放器，本文正是在参考目前市面上的播放器后，通过 Qt 开发出一款简单易用的播放器。

这款播放器相较于那些功能繁多的播放器来说拥有更加简洁纯粹的用户界面方便用户使用，同时本身体积小，运行所占内存也小，同时因为它是用 Qt 编写的源代码，只要一次编写就可以在拨通的平台上编译运行不需要修改源代码。

第二章　相关技术研究

2.1 开发平台以及开发语言介绍介绍

2.1.1Qt 平台介绍

Qt 是一个跨平台 C++图形用户界面应用程序开发框架。能够非常轻松的实现“一次编写，随处编译”的功能。这让我们编写也得应用程序可以非常轻易的在各大系统平台上完美运行，从而大大提高了代码的复用性

Qt 提供信号和插槽概念，这是一种类型安全的方式，它可以回调，同时也支持对象之间在彼此不知道对方信息的情况下进行协同工作，这使得 Qt 非常适合于真正的构件编程

Qt 提供了大量的帮助和参考文档，其中包括了超文本 HTML 方式，还有 Unix 帮助页 man 手册页以及补充说明。对于初学者们来说，这对于他们快速学习使用 Qt 提供了非常大的帮助。

Qt Creator 是一个全新的跨平台集成开发环境，它可以单独使用，也可以将它和 Qt 库以及其他的开发工具与结合起来使用。Qt Creator 编译器支持 C++代码的编译上下文代码的相关帮助、代码的自动补齐等功能。

Qt Creator 中的个文件介绍

.pro 文件是 qmake 的工程文件，这个文件主要用来设置编译或者是连接的变量。其中主要包括了 TEMPLATE 用于确定该项目最终被编译成什么形式；TARGET 定义了最后生成的名字；DEPENDPATH 设置了该工程所需要依赖的路径；SOURES 定义了该项目所需的源文件；LIBS 定义了项目用到的动态数据库等等

.h 文件是用来存放存放函数声明的头文件，编译的时候会将其功能具体是现在.cpp 文件中。

.cpp 文件是项目工程的实现文件，是一个项目的主要构成，主要用于实现在头文件中声明过的方法。

.ui 文件用于编辑 C++可视化界面，它不能直接用代码进行修改，需要通过 Qt

Desinger 工具进行修改，界面类似于我们常用的 VB，而且还提供了大量可供编程使用的组件。

.qrc 文件是一个 xml 格式的资源配置文件，用来记录指点的资源文件。

2.1.2C++语言介绍

C++是 1979 年在新泽西州美利山贝实验室中被研发出来的，它的开发着是 Bjarne Stoustrup。C++可以说是一种对 C 语言进行拓展和完善的一种语言。它是一款完全面向对象的编程语言，支持封装、抽象、多态和继承。

C++的主要特征有：对类型的检查更加严格；增加了异常处理的机制；有自己的标准模板库；是一种面向对象的编程语言；增加了运算符的重载等。

C++的优势在于高端和底层开发还是以 C/C++为主，它的执行效率非常高，在各个平台上代码级的移植性非常高，能够轻松的实现跨平台。

第三章 播放器功能设计和实现

3.1 系统简介

该播放器可以再 Windows 平台、Linux 平台等平台上播放各种格式的多媒体文件。同时也可以播放有服务器的流媒体视频软件。无论在什么平台下运行手不需要修改源代码。本播放器的使用方法与大多数的播放器一样，拥有简洁的界面，易于操作的控制按钮。可以进行视频文件的播放、暂停、播放进度的调整。本地文件的打开播放以及服务器端网络文件的播放。

3.2 功能需求描述

1、能够打开本地文件夹寻找到本地的媒体文件进行播放。

2、本地媒体播放包含了音频文件、视频文件的浏览、播放。

3、能够获取到服务器的目录并且能够播放服务器中的文件。

4、流媒体播放包含了音频文件、视频文件的浏览、播放。

5、能够实现播放媒体文件的声音控制。

6、能够实现播放媒体文件的进度控制。

7、能够实现播放媒体文件的速度控制。

3.3 整体系统结构设计

跨平台播放器系统主要包括了两大模块。首先是播放器的客户端，能够实现本地媒体文件的播放，同时也能够获取到服务器端的文件目录，并且进行播放。第二模块是服务器模块,主要用来存放流媒体文件,并且通过 HTTP 协议传输文件，实现流媒体文件在客户端的播放。对于一个多媒体播放器来说,用户最基本的要求就是播放多媒体文件。用户在观看视频时会根据自身实际需求来进行播放对象的一些调整如控制视频播放和暂停、调整视频音量、调整播放进度、调整播放速度等。同时播放器也要支持多种格式的视频播放。

系统整体结构图如图 1 所示。

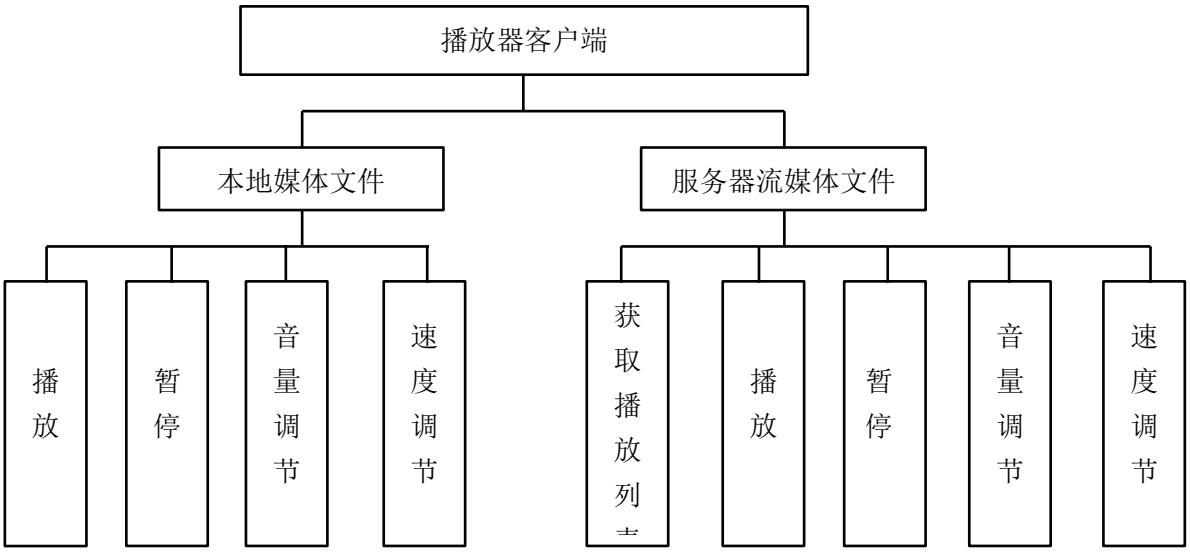

图 1 系统功能模块图

播放器与服务器的请求流程图如图 2 所示。

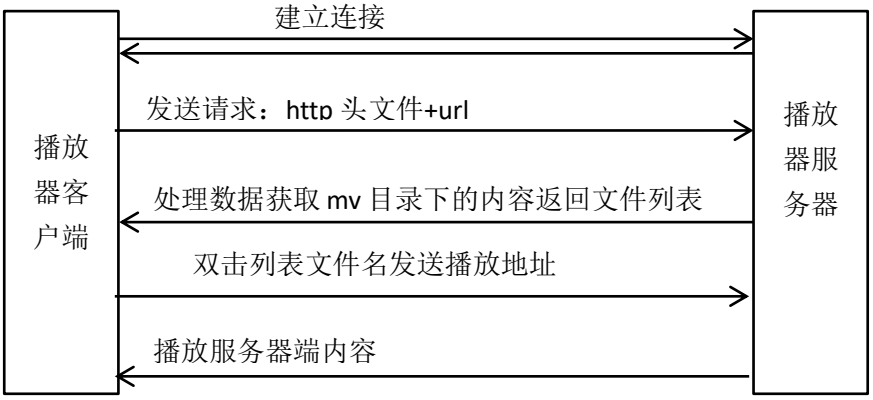

图 2 播放器与服务器请求流程图

3.4 播放器客户端的界面设计

3.4.1 播放器主界面设计

利用 Qt Creator 自带的 Qt Desinger 工具进行播放器主界面的设计；利用 List Widget 控件实现对服务器上播放文件列表的展示；利用 Widget 控件将其提升为 QVideoWidget 实现视频的播放功能；利用 Tool Button 控件实现对播放视频文件的打开、播放、暂停、静音等功能的实现；利用 Label 控件实现播放时间的显示；利

用 Horizontal Slider 控件实现对播放进度的展示和调节功能以及音量的控制。

主界面设计图如图 3。

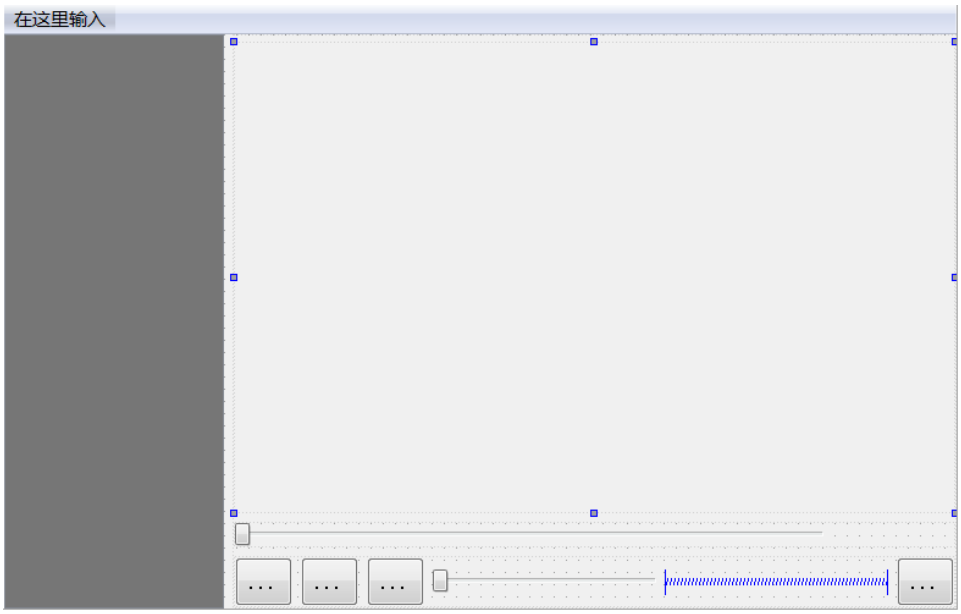

图 3 主界面设计图

主界面背景设置：

```
QPixmap pixmap(":/images/background.jpg");
QPalette palette;
palette.setBrush(backgroundRole(),QBrush(pixmap));
setPalette(palette);
```

播放按钮图标设置：

```
ui->toolButton->setToolTip("播放");
ui->toolButton->setIcon(QPixmap(":/images/play.png"));
```

暂停按钮图标设置：

```
ui->toolButton->setToolTip("暂停");
ui->toolButton->setIcon(QPixmap(":/images/pause.png"));
```

打开文件按钮图标设置：

ui->toolButton_2->setToolTip("打开文件");

ui->toolButton_2->setAutoRaise(true);

ui->toolButton_2->setIcon(QPixmap(":images/openfile.png"));

静音按钮图标设置：

ui->toolButton_3->setToolTip("静音");

ui->toolButton_3->setIcon(QPixmap(":images/voice.png"));

设置按钮图标设置：

ui->toolButton_4->setToolTip("设置");

ui->toolButton_4->setAutoRaise(true);

ui->toolButton_4->setIcon(QPixmap(":images/set.png"));

主界面运行图如图 4。

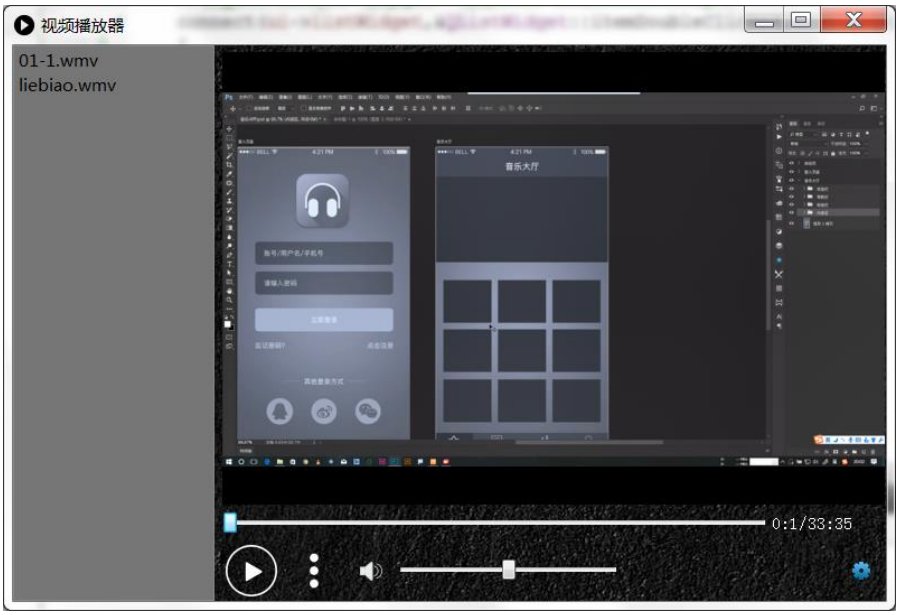

图 4 主界面运行图

3.4.2 播放器子界面设计

利用 Qt 设计界面类添加子界面窗体，使用 Qt Desinger 设计子界面的整体风格。

子界面设计图 5 所示。

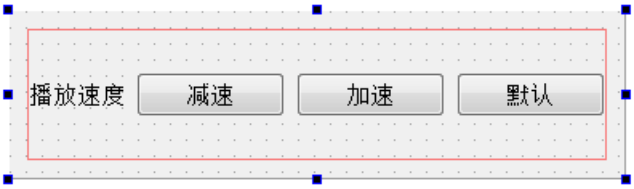

图 5 子界面设计图

子界面设计 xml 文件

```
//窗体名称设置
<widget class="QDialog" name="SetDialog">
   <property name="windowTitle">
    <string>播放速度设置</string>
   </property>
</widget>
//Label 控件 text 属性设置
<widget class="QLabel" name="lab_speed">
  <property name="text">
   <string>播放速度</string>
  </property>
</widget>
//减速按钮 text 属性设置
<widget class="QPushButton" name="btn_down">
  <property name="text">
   <string>减速</string>
  </property>
</widget>
//加速按钮 text 属性设置
<widget class="QPushButton" name="btn_up">
  <property name="text">
```

```
  <string>加速</string>
 </property>
</widget>
//默认速度按钮 text 属性设置
<widget class="QPushButton" name="btn_normal">
 <property name="text">
  <string>默认</string>
 </property>
```

子界面运行图如图 6 所示。

图 6 子界面运行图

3.5 播放器程序文件组成

该播放器的源代码文件组成包括了 VideoPlayer.pro 文件、头文件、源文件、界面文件、资源文件。文件结构如图 7 所示。

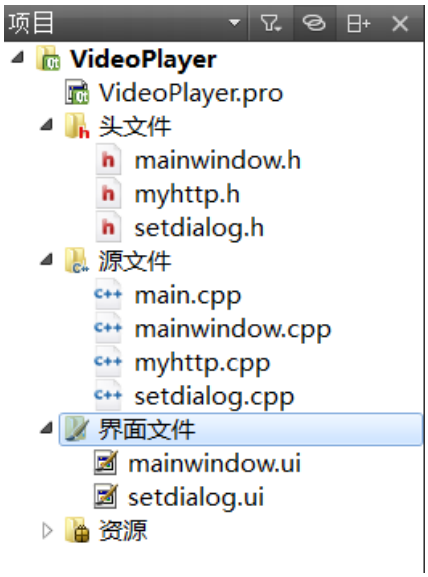

图 7 源程序文件结构图

3.5.1VideoPlayer.pro 文件的功能

该文件是 Qt 的项目文件，在该文件中指明了工程所需要使用的 Qt 模块，其中包括了 core gui 模块用来编辑可视化界面；multimedia 模块用来提供多媒体的功能，包括文件大小，声音控制等；multimediawidgets 模块用来提供多媒体播放窗口；network 模块提供该程序所需要的网络服务相关功能。

同时他也定义了该工程的名字以及工程的类型代码如下：

TARGET = VideoPlayer　　//工程名 VideoPlayer

TEMPLATE = app　//该工程类型为 app 形式

在该文件中指明了工程所包含的源文件、头文件和界面文件代码如下：

```
SOURCES += main.cpp\
           mainwindow.cpp \
           myhttp.cpp \
           setdialog.cpp
HEADERS   += mainwindow.h \
           myhttp.h \
           setdialog.h
FORMS      += mainwindow.ui \
           setdialog.ui
```

3.5.2 头文件中包含和功能

（1）mainwindow.h 文件定义了所有在主窗体中所需要使用的类名、方法名、信号。代码如下:

```
private:
    Ui::MainWindow *ui;
    QMediaPlayer *mediaPlayer;//QVideoWidget
    bool flag;
    QTimer *timer;
    qint64 stime;
    QString totaltime, currenttime;
    double_t speed;
private slots:
    //播放
    void play();
    //播放状态
    void mediaStateChanged(QMediaPlayer::State state);
    //改变播放位置
    void positionChanged(qint64 position);
    //获取播放位置

    void setPosition(int position);
```

```
        //播放长度
        void durationChanged(qint64 position);
        //获取服务器的文件列表
        void getFile();
        //创建 listwidget 条目 item   添加条目名称
        void createItem(QString item);
        void on_toolButton_clicked();
        void on_horizontalSlider_sliderMoved(int position);
        void on_toolButton_2_clicked();
        void on_toolButton_3_clicked();
        void on_voice_valueChanged(int value);
        void on_toolButton_4_clicked();
        void timerUpDate();
        void SpeedDown();
        void SpeedUp();
        void SpeedNormal();
    signals:
        void setdlgshow();
```

（2）myhttp.h 文件定义了连接服务器的功能方法名以及获取文件列表的方法名，代码如下：

```
public:
myhttp();
```

//获取网络中文件列表 url 地址 port 端口

QString getFileList(QString URL,int port);

（3）setdialog.h 文件定义了子窗体的类名、需要发送的信号和接受的信号以及子窗体中的所有方法名，代码如下：

```
private:
    Ui::SetDialog *ui;
private slots:
    void SetDlgShow();
    void on_btn_down_clicked();
    void on_btn_up_clicked();
    void on_btn_normal_clicked();
signals:
    void speedDown();
    void speedUp();
    void speedNormal();
```

3.5.3 源文件内容和功能

（1）main.cpp 文件是该程序的主程序的源文件，主要实现了主窗体和子窗体的初始化和两个窗体之间的功能交互实现了在主窗体上点击设置按钮弹出子窗体，在子窗体上点击相应按钮实现播放的速度变化。代码如下：

当主窗体发送 setdlgshow()信号时子窗体执行 SetDlgShow()方法用来显示子窗体

QObject::connect(&w,SIGNAL(setdlgshow()),&dlg,SLOT(SetDlgShow()));

当子窗体发送 speedDown()信号的时候主窗体执行 SpeedDown()方法来降低播放速度

QObject::connect(&dlg,SIGNAL(speedDown()),&w,SLOT(SpeedDown()));

当子窗体发送 speedUp ()信号的时候主窗体执行 SpeedUp ()方法来提高播放速度

QObject::connect(&dlg,SIGNAL(speedUp()),&w,SLOT(SpeedUp()));

当子窗体发送 speedNormal ()信号的时候主窗体执行 SpeedNormal ()方法来恢复播放速度

```
QObject::connect(&dlg,SIGNAL(speedNormal()),&w,SLOT(SpeedNormal()));
```

（2）setdialog.cpp 文件中实现了子窗体接收信号显示子窗体和给主窗体发送信号的功能，代码如下：

```
void SetDialog::SetDlgShow()
{
    //接收信号后显示子窗体
    this->show();
}
void SetDialog::on_btn_down_clicked()
{
    //当减速按钮被点击时发送 speedDown()的信号
    emit speedDown();
}
void SetDialog::on_btn_up_clicked()
{
    //当加速按钮被点击时发送 speedUp ()的信号
    emit speedUp();
}
void SetDialog::on_btn_normal_clicked()
{
   //当正常按钮被点击时发送 speedNormal ()的信号
    emit speedNormal();
}
```

（3）myhttp.cpp 文件主要实现了播放器客户端连接服务器以及从服务器端获

取文件列表的功能，代码如下：

```
QString myhttp::getFileList(QString URL, int port)
{
    QHostAddress serverIP;
    serverIP.setAddress(URL);
    //连接到 server 端
    connectToHost(serverIP,port);
    //等待连接
    waitForConnected();
    //获取 http 格式化的请求
    char buf[1024];
    memset(buf,0,sizeof(buf));
    //将格式化的字符串放在 buf 中     格式 http 表头文件+ url 地址
    sprintf(buf,REQ,URL.toStdString().data());
    //阻塞等待 TCP 写状态 OK
    waitForBytesWritten();
    //将 HTTP 请求发送到 server 端
    write(buf,strlen(buf));
    //阻塞等待数据传输到本地
    waitForReadyRead();
    //保存从 server 端接收到的数据
    QString content;
    while(bytesAvailable()>0)
    {
        memset(buf,0,sizeof(buf));
        read(buf,sizeof(buf));
        content+=buf;
```

```
    }
    //等待阻塞断开完成
    //waitForDisconnected();
    //完成之后关闭 socket
    close();
    //处理 content 数据\r\n 分割 content
    QStringList ls = content.split("\r\n");
    return ls[ls.count()-1];
}
```

（4）mainwindow.cpp 文件实现了播放器的所有功能其中包括了：

媒体文件播放和暂停功能实现：

利用 Tool Button 控件自动转化为槽函数，完成播放按钮的功能。并且通过监听播放状态实现一个按按钮同时实现两种功能。

设置播放属性代码：

1、监听播放信号变化函数，当监听到 mediaPlayer 的 stateChanged(QMediaPlayer::State) 函数变化时当前窗体执行 mediaStateChanged(QMediaPlayer::State)操作，实现 mediaplay 播放过程中动态调整播放进度，代码如下：

```
connect(mediaPlayer,SIGNAL(stateChanged(QMediaPlayer::State)),this,SLOT(mediaStateChanged(QMediaPlayer::State)));
```

2、播放进度信号变化函数，当监听到 mediaPlayer 的 positionChanged(qint64) 函数变化时当前窗体执行 positionChanged(qint64)操作，用来改变进度条位置，从而调整播放进度，代码如下：

```
connect(mediaPlayer,SIGNAL(positionChanged(qint64)),this,SLOT(positionChanged(qint64)));
```

3、播放长度信号变化当监听到 mediaPlayer 的 durationChanged(qint64)函数变

化时当前窗体执行 durationChanged(qint64)操作，用来设置进度条大小，代码如下：

```
connect(mediaPlayer,SIGNAL(durationChanged(qint64)),this,SLOT(durationChanged(qint64)));
```

获取播放状态代码：

```
switch (mediaPlayer->state()) {
    case QMediaPlayer::PlayingState:
        mediaPlayer->pause();
        break;
    default:
        mediaPlayer->play();
        break;
    }
```

播放和暂停按钮功能转换，当媒体文件在播放时播放按钮背景改变为 pause.png，鼠标悬停提示改变为“暂停”，同时隐藏播放列表列表；当媒体文件暂停时播放按钮背景改变为 play.png，鼠标悬停提示改变为“播放”，同时显示播放列表列表。

```
switch (state) {
    case QMediaPlayer::PlayingState:
        ui->listWidget->hide();
        ui->toolButton->setToolTip("暂停");
        ui->toolButton->setIcon(QPixmap(":/images/pause.png"));
        break;
    default:
        ui->listWidget->show();
        ui->toolButton->setToolTip("播放");
        ui->toolButton->setIcon(QPixmap(":/images/play.png"));
        break;
```

}

打开本地媒体文件功能实现：

利用 Tool Button 控件自动转化为槽函数，实现打开本地文件夹的功能。并且利用 QFileDialog::getOpenFileName()函数获取媒体文件的播放路径从而实现本地媒体文件的播放。

打开按钮功能，首先定义一个字符串 filename，将获取到的文件名及文件路径保存在字符串中，：设置播放文件的本地地址为 filename，代码如下：

```
QString filename = QFileDialog::getOpenFileName();
 //设置播放内容
 mediaPlayer->setMedia(QUrl::fromLocalFile(filename));
 ui->toolButton->setAutoRaise(true);
```

显示当前播放时间和总播放时长功能实现：

利用 Label 控件的 setText() 方法和 QMediaPlayer->position() 方法和 QMediaPlayer-> duration ()方法获取到的当前播放时间和总播放时间，从而实现显示当前播放时长和总播放时长的功能。同时利用 QTimer()方法实现每隔 0，1 秒刷新一次的功能。

获取当前媒体文件已播放代码：

```
stime=mediaPlayer->position();                  // 获取当前已播放时长（毫秒）
qint64 time=stime/1000;                         //获取当前前已播放时长（秒）
qint64 minute = time / 60;                        //获取当前前已播放时长（分）
int second = time % 60;          //获取当前前已播放时长除去分钟数的秒数（秒）
currenttime = QString::number(minute) + ":" + QString::number(second);
//将获得的分秒放在一个字符串中
```

获取当前媒体文件的总时长代码：

```
qint64 minute;
```

```
int second;
qint64 time;
time = mediaPlayer->duration()/1000;          //得到总共的秒数
minute = time/60;                             //获得分
second = time%60;                             //获得秒
totaltime = QString::number(minute) + ":" + QString::number(second);
```

//将获得的分秒放入 totaltime 字符串中

利用 Label 实现播放时长和总时长显示的代码：

```
QString Time = currenttime + '/' + totaltime;
ui->lab_nowtime->setText(Time);
```

每隔 0.1 秒刷新一次的代码：

```
timer = new QTimer(this);
connect(timer,SIGNAL(timeout()),this,SLOT(timerUpDate()));
timer->start(100);                    //设置刷新时间为 0.1 秒
```

媒体文件播放位置改变功能实现：

使用 Horizontal Slider 控件完成该功能的实现。利用 QMediaPlayer-> duration () 方法获取到当前文件的总时长并且使用 setRange()方法完成对进度条数值长度的设置，通过监听进度条数值的改变来实现播放进度的改变，同时也通过监听播放进度的改变来改变进度条的数值。

首先通过监听 mediaPlayer 对象的 durationChanged(qint64)函数变化来设置进度条的长度信号和槽函数代码如下：

```
connect(mediaPlayer,SIGNAL(durationChanged(qint64)),this,SLOT(durationChanged(qint64)));
void MainWindow::durationChanged(qint64 duration)
{
    //设置进度条的范围
```

```
    ui->horizontalSlider->setRange(0,duration);
}
```

然后通过 on_horizontalSlider_sliderMoved(int position)函数获取到进度条改变后的播放时间并且利用 setPosition(int position)函数设置播放进度，代码如下：

```
void MainWindow::on_horizontalSlider_sliderMoved(int position)
{
    setPosition(position);
}
void MainWindow::setPosition(int position)
{
    //获取 mediaplayer 进度条调整位置
    mediaPlayer->setPosition(position);
}
```

静音功能实现 ：

使用 Tool Button 控件自动转换槽函数，利用 bool 类型判断来改变媒体文件播放时声音的打开和关闭。

声音开关功能实现，首先设置 flag 为 bool 类型并且默认为 true，点击按钮判断当 flag 为真时设置 mediaPlayer 对象的 Volume 属性为 false 实现静音功能，同时改变按钮背景图片为 voice2.png，并将 flag 设为 false，当再次点击按钮时，这时的 flag 属性为 false，则将 mediaPlayer 对象的 Volume 属性设置为 ui->voice 对象的当前设置，实现声音的打开，同时改变背景图片为 voice.png，代码如下：

```
if(flag)
{
    mediaPlayer->setVolume(false);
    ui->toolButton_3->setIcon(QPixmap(":images/voice2.png"));
    flag=false;
```

```
}
else
{
    mediaPlayer->setVolume(ui->voice->value());
    ui->toolButton_3->setIcon(QPixmap(":images/voice.png"));
    flag=true;
}
```

控制声音大小功能实现：

利用 Horizontal Spacer 控件绑定 mediaPlayer->setVolume()属性，通过改变 Horizontal Spacer 值的大小来实现对声音的控制。

控制播放声音大小，代码如下：

```
void MainWindow::on_voice_valueChanged(int value)
{
    mediaPlayer->setVolume(ui->voice->value());
}
```

控制播放速度功能实现：

利用 Qt 特有的信号和槽机制，通过监控 Tool Button 控件的单击事件来改变 mediaPlayer->setPlaybackRate()方法的值来改变播放速度。

减速播放功能实现代码：

```
void MainWindow::SpeedDown()
{
    //实现了每次单击按钮都可以减速为当前速度的 0.5 倍
    speed=speed/2;
    mediaPlayer->setPlaybackRate(speed);
}
```

加速播放功能实现代码：

```
void MainWindow::SpeedUp()
{
    //实现了每次单击按钮都可以加速为当前速度的 2 倍
    speed=speed*2;
    mediaPlayer->setPlaybackRate(speed);
}
```

正常速度播放实现代码：

```
void MainWindow::SpeedNormal()
{
    //正常速度播放文件
    mediaPlayer->setPlaybackRate(1);
}
```

显示流媒体文件列表功能实现

使用 List Widget 控件，通过添加 item 的方法将所有文件名添加到列表中。

获取文件名并且添加到列表中代码：

```
void MainWindow::getFile()
{
    myhttp http;
    QString content = http.getFileList("192.168.43.58",80);
    qDebug()<< content;
    //获取到的服务器资源列表
    //将资源列表中的内容 变成 listwidget 中的条目  item
    QStringList ls = content.split("\n");
    for(int i=0;i<ls.count();i++)
    {
```

```
            //qDebug()<<ls[i];
            //如果 ls[i]不为空 将 ls[i]添加到 item 中
            if(!ls[i].isEmpty())
            {
                createItem(ls[i]);
            }
        }
    }
```

双击列表播放流媒体文件功能实现：

设置 listwidget 处理事件和相应 item doubleClicked 事件代码：

```
    connect(ui->listWidget,&QListWidget::itemDoubleClicked,this,[=](QListWidgetIte
m * item)
        {
            //获取播放的 url 地址，根据地址打开播放的文件
            QUrl url;
            url.setUrl("http://192.168.43.58:80/mv/"+item->text());
            //设置 mediaplayer 的播放内容
            mediaPlayer->setMedia(url);
            play();
        });
```

3.6 播放器服务器功能实现

3.6.1 获取服务器端的列表功能实现

获取服务器下 mv 文件夹中的媒体文件列表代码：

```
int main(int argc, char *args[])
```

```
{
    DIR *dp;
    struct dirent *dirp;
    dp = opendir("./mv");
    if (dp == NULL)
    {
        //printf("");
        return 0;
    }
    char buf[2048] = { 0 };
    while((dirp = readdir(dp)) != NULL)
    {
        char tmp[100] = { 0 };
        if (strcmp(dirp->d_name, ".") != 0 && strcmp(dirp->d_name, "..") != 0)
        sprintf(tmp, "%s\n", dirp->d_name);
        strcat(buf, tmp);
    }
    closedir(dp);
    printf("%s\n", buf);
    return 0;
}
```

从路径中得到文件名代码：

```
const char *getfilename(const char *path)//从完整路径中得到文件名
{
    int i;
    for(i = strlen(path); i >= 0; i--)
    {
        if (path[i] == '\\' || path[i] == '/')
            return &path[i + 1];
    }
    return path;
}
```

从路径中得到文件拓展名代码：

```
void get_file_ext(const char *filename, char *extname)//得到文件扩展名
{
    int len = strlen(filename);
    int i;
    for (i = len - 1; i >=0; i--)
    {
        if (filename[i] == '.')
        {
            strncpy(extname, &filename[i + 1], 1023);
```

```
                break;
            }
        }
    }
```

根据参数 port，建立 server 端 socket 代码：

```
int socket_create(int port) //根据参数 port，建立 server 端 socket
{
    int st = socket(AF_INET, SOCK_STREAM, 0); //建立 TCP 的 socket 描述符
    if (st == -1)
    {
        printf("socket failed %s\n", strerror(errno));
```

```
        return 0;
    }
    int on = 1;
    if (setsockopt(st, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on)) == -1)
    {
        printf("setsockopt failed %s\n", strerror(errno));
        return 0;
    }
    struct sockaddr_in addr;
    memset(&addr, 0, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(port);
    addr.sin_addr.s_addr = htonl(INADDR_ANY);
    if (bind(st, (struct sockaddr *) &addr, sizeof(addr)) == -1)
    {
        printf("bind failed %s\n", strerror(errno));
        return 0;
    }
    if (listen(st, 200) == -1)
    {
        printf("listen failed %s\n", strerror(errno));
        return 0;
    }
    printf("listen %d success\n", port);
    return st; //返回 listen 的 socket 描述符
}
```

3.6.2 从服务器端获取请求后的处理

从 http 请求中读出 GET 后面的命令行代码：

```
int i;
int istart = 0;
int iend = 0;
int msg_len = strlen(sHTTPMsg);
for (i = 0; i < msg_len; i++)
{
    if ((sHTTPMsg[i] == ' ') && (istart == 0))
    {
        istart = i + 2;
    }
    else
    {
        if (sHTTPMsg[i] == ' ')
        {
            iend = i;
            break;
        }
    }
}
```

```
int len = iend - istart;
if (istart > msg_len || len <= 0 || len >= 2048)
    return;
strncpy(command, &sHTTPMsg[istart], len);
```

得到 post 的长度和内容代码：

```
const char *s = strstr(sHTTPMsg, POST_LEN);
if (s)
{
    int end = 0;
    while (s[end] != '\n' && s[end] != '\0')
    {
        end++;
    }
    char len[100] = { 0 };
    strncpy(len, &s[strlen(POST_LEN)], end);
    int i_len = atoi(len);
    s = strstr(s, "\r\n\r\n");
    if (s)
    {
        *content = malloc(i_len);
        memcpy(*content, &s[4], i_len);
        return i_len;
    }
}
return 0;
```

得到 post 消息 head 的长度代码：

```
int index = 0;
while (sHTTPMsg[index + 3] != '\0')
{
    if (sHTTPMsg[index] == '\r' && sHTTPMsg[index + 1] == '\n' &&
        sHTTPMsg[index + 2] == '\r' && sHTTPMsg[index + 3] == '\n')
        break;
    index++;
}
return index + 4;
```

3.6.3 服务器响应机制实现

当服务器 mv 文件夹中没有文件时函数返回 0，当 mv 文件夹中有文件时则建立 socket 连接，并且设置 socket 状态为 daemon，同时捕捉客户端发出的信号，当有 hppt 的请求服务器就会进行相应事件处理：

```
if (arg < 2)//如果没有参数，main 函数返回
{
    printf("usage:%s port\n", args[0]);
    return EXIT_FAILURE;
}
int iport = atoi(args[1]);//将第一个参数转化为整数
if (iport == 0)
{
    printf("port %d is invalid\n", iport);
    return EXIT_FAILURE;
}
if (arg >= 3)
    allow_postfile = atoi(args[2]);
int st = socket_create(iport);//建立 listen socket
if (st == 0)
    return EXIT_FAILURE;
```

```
printf("myhttp begin\n");
writelog("myhttp begin");
setdaemon();//设置进程为 daemon 状态
signal1(SIGINT, catch_Signal);//监控 SIGINT 信号
signal1(SIGPIPE, catch_Signal);
signal1(SIGALRM, catch_Signal);
signal1(SIGTERM, catch_Signal);
socket_accept(st);
close(st);
printf("myhttp end\n");
writelog("myhttp end");
return EXIT_SUCCESS;
```

第四章 系统测试

4.1 测试环境与配置

测试环境：

播放器客户端：windows 7 旗舰版系统

服务器：VMware Workstation Pro 搭载 Ubuntu64 位系统

4.2 测试方法

手动利用和黑盒测试方法进行测试，主要测试目标为是否能够成功运行程序。

4.3 测试结果

测试执行情况与记录如表 1：

表 1 测试记录表

测试用例编号	测试过程	功能概述	测试结果
1	打开本地文件	使用打开文件按钮找到本地文件并且将其打开	通过
2	播放视频文件	使用播放按钮播放本地视频文件	通过
3	暂停视频文件	使用暂停按钮暂停本地视频文件	通过
4	播放音频文件	使用播放按钮播放本地音频文件	通过
5	暂停音频文件	使用暂停按钮暂停本地音频文件	通过
6	播放流媒体文件	通过双击列表中的文件名实现对流媒体文件的播放	通过
7	暂停流媒体文件	使用暂停按钮暂流媒体文件视频文件	通过
8	关闭声音	通过声音按钮实现静音功能	通过

9	打开声音	通过声音按钮实现打开功能	通过
10	改变音量大小	通过拖动声音控制进度条的滑块改变声音的大小	通过
11	改变播放速度	通过不同的按钮实现视频播放的加速、减速、正常速度播放	通过
12	改变播放进度	通过拖动播放进度条的滑块改变播放进度	通过
13	显示播放时长和总时长	使用 label 控件显示已播放进度的分钟秒数以及总的分钟秒数	通过

第五章 使用说明

整个软件的使用方法如下：

将 Ubuntu 系统作为此播放器的服务器，将 http-server-cgi 文件夹传到 Ubuntu 系统中，开启 Ubuntu 系统的 SSH 服务代码如下：

sudo apt-get update

sudo apt-get install openssh-server

进入 http-server-cgi 文件夹编译整个文件代码如下：

cd http-server-cgi

sudo make

启动服务代码如下：

sudo ./myhttp start

手动打开软件运行，在 windows 环境下可以直接方便的打开播放器，不需要安装和调试，实现了让用户方便快速使用，使得软件的运行效率更高。

打开文件查找需要播放的文件，使用 windows 的选择文件窗口选中要打开的文件，点击确定即可打开。

文件播放，点击播放按钮实现播放和暂停功能，双击播放列表，播放流媒体文件。

控制音量，用户可以根据自身需求选择合适的音量播放。

控制播放进度，提高用户体验，方便用户根据自身需求选择播放的进度。

控制播放速度，进一步提高用户体验，方便用户根据自身需求选择播放的速度。

第六章 结束语

总结

致谢

参考文献

[1]布兰切特.C++GUI Qt 4 编程[M].电子工业出版社，2013.

[2]吴迪.零基础学 Qt4 编程[M].北京航空航天大学出版社，2010.

[3]刘晓立，赵俊逸。基于 Qt 音乐播放器[J].软件导刊，2015，14（10）：112-114.

[4] Jasmin Blanchette And Mark Summer field.C++GUI Program-mingwithQT4[M].第二版.电子工业出版社，2008.

[5]霍亚飞.Qt Creator 快速入门[M].北京航空航天大学出版社,2012.

[6]王建民.基于 Qt 的嵌入式媒体播放器系统的设计[C].微机计算机信息,2000,17(1):84-86.