

课程设计

课程名称： 嵌入式系统课程设计

设计名称： MP3播放器的设计与实现

专业班级： 物联网1502 学号： 2015003588

学生姓名： 刘港

指导教师： 兰方鹏

2018年6月20日

太原理工大学课程设计任务书

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学生姓名 | 刘港 | 专业班级 | 物联网工程1502 | |
| 课程名称 | 嵌入式系统课程设计 | | | |
| 设计名称 | MP3播放器的设计与实现 | | 设计周数 | 2周 |
| 设计  任务  主要  设计  参数 | 1、制作一个可以在Linux环境下运行的基于madplay库的MP3播放器，实现音乐的播放、暂停、继续、停止、上一曲、下一曲等功能；  2、对madplay源码进行移植，安装交叉编译工具链，搭建交叉编译环境，使得该程序可以在ARM平台上运行  3、在Linux平台（Redhat或Ubuntu）环境下配置QT环境，设计一个基于图形化界面的MP3播放器。 | | | |
| 设计内容  设计要求 | 1、第一部分的程序运行时显示字符界面，在该字符界面下输入相应的提示字符即可实现音乐的播放、暂停、继续、停止、上一曲、下一曲等功能；编写makefile文件，make实现程序的编译  2、第三部分的界面实现音乐的播放、暂停、继续、停止、上一曲、下一曲、歌词的显示等功能 | | | |
| 主要参考  资料 | 《GNU/LINUX编程》（郑谦益编著）  CSDN博客 | | | |
| 学生提交  归档文件 | 封面—任务书—目录—课程设计报告 | | | |

注：

1.课程设计完成后，学生提交的归档文件应按照：封面—任务书—说明书—图纸的顺序进行装订上交（大张图纸不必装订）。

2.可根据实际内容需要续表，但应保持原格式不变。

**指导教师签名**： **日期**：2018年6月20日

目录

[一、【问题描述】 1](#_Toc518810359)

[二、【设计需求及分析】 1](#_Toc518810360)

[1、背景简介 1](#_Toc518810361)

[2、QT 界面主程序流程图 2](#_Toc518810362)

[三、【设计功能的实现】 2](#_Toc518810363)

[1、Linux环境下安装QT 2](#_Toc518810364)

[2、QT实现MP3 播放器 4](#_Toc518810365)

[2.1在QT Creator创建MP3播放器的QT项目 4](#_Toc518810366)

[2.2 MP3播放器的ui界面的设计 6](#_Toc518810367)

[2.3 UI相关控件槽函数的设计 6](#_Toc518810368)

[2.4 UI界面的美化 10](#_Toc518810369)

[2.5成员变量和头文件说明 11](#_Toc518810370)

[四、【运行结果】 11](#_Toc518810371)

[1、Qt MP3播放器的运行 11](#_Toc518810372)

[五、【心得体会】 14](#_Toc518810373)

[六、【附录】 15](#_Toc518810374)

[1、widget.cpp源代码 15](#_Toc518810375)

[2、main.cpp源码 22](#_Toc518810376)

**MP3播放器的设计与实现**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

专业： 物联网工程 班级： 1502 姓名：刘港 学号：2015003588 完成日期：2018-07-06 完成模块：QT MP3播放器的实现

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

# 一、【问题描述】

⑴ 制作一个可以在Linux环境下运行的MP3播放器。要求实现的MP3主要功能包括：播放、暂停、继续、停止、循环播放、上一曲、下一曲、退出、歌曲显示等功能。

本设计是基于madplay库实现的，程序运行时会显示一个字符界面，在该字符界面下输入相应的提示字符即可实现音乐的播放、暂停、继续、停止、上一曲、下一曲、歌词的显示等功能，也可以加入额外的一些功能。编写一个MakeFile文件，make命令来实现程序的编译。

⑵ 对madplay源码进行移植，安装交叉编译工具链，搭建交叉编译环境，使得该程序可以在ARM平台上运行。

⑶ 在Linux平台（Redhat或Ubuntu）环境下配置QT环境，设计一个基于图形化界面的MP3播放器。

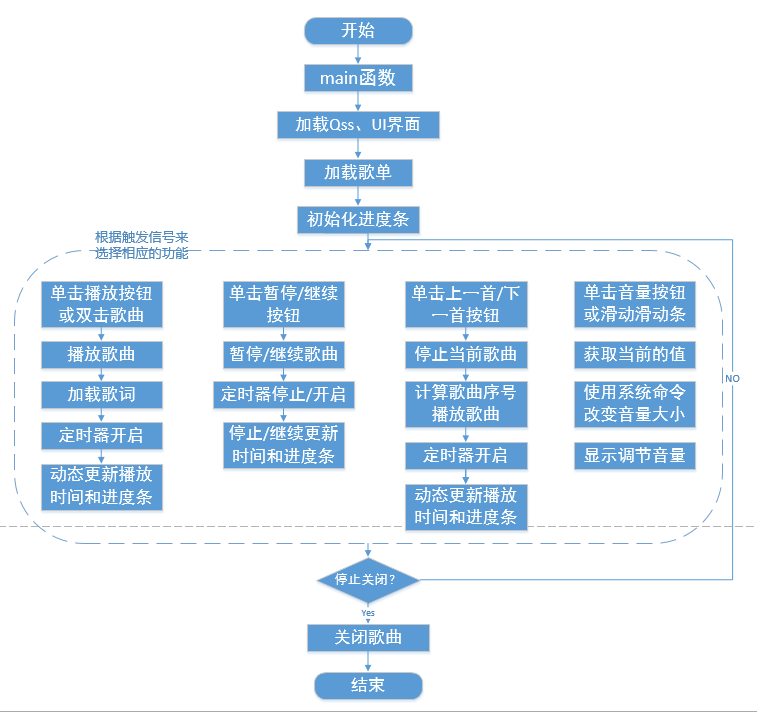
# 二、【设计需求及分析】

## 1、背景简介

随着嵌入式行业的日新月异,嵌入式系统的应用越来越受人喜爱,自由可移植的优点促使人们更倾向于关注嵌入式技术的研究,媒体类的产品被人们广泛而快速的使用,尤其是音乐播放器的软件开发成为了众多企业争宠的对象。

Madpaly 是 linux 上的一个开源 MP3 播放器,是一个高精度 MPEG 音频解码器。支持MPEG-1 Layer,Layer II 和 Layer III(也就是 MP3)标准,利用 libmad 库进行 MP3 解码。本课程设计利用 Madplay 库实现一个简单的音乐播放器,可以播放本地的音频文件并具备基本的音乐操作处理功能,播放暂停恢复停止音量操作等。

## 2、QT 界面主程序流程图



二 图2 1

# 三、【设计功能的实现】

注：系统采用ubuntu16.04

## 1、Linux环境下安装QT

Qt 是一个跨平台的 C++应用程序框架,支持 Windows、Linux、Mac OS X、Android、iOS、Windows Phone、嵌入式系统等。即 Qt 可以同时支持桌面应用程序开发、嵌入式开发和移动开发,覆盖了现有的所有主流平台。

这次课程设计我们在 QT 平台上开发一个 MP3 播放器,首先需要在 Linux 平台上安装并配置 QT 开发环境,配置安装如下。

**1.1 下载 QT 源码软件,并解压**

http://download.qt.io/archive/qt/4.7/qt-everywhere-opensource-src-4.7.3.tar.gz.mirrorlist

我们统一下载的版本连接为:

http://mirrors.tuna.tsinghua.edu.cn/qt/archive/qt/4.7/qt-everywhere-opensource-src-4.7.3.tar.gz

**1.2 执行 QT 配置**

进入 qt-everywhere-opensource-src-4.7.3 目录,执行配置文件 configure

Ubuntu 环境下需要超级账号执行,即:sudo ./configure

当出现系统出现以下信息时:

Which edition of Qt do you want to use ?

Type 'c' if you want to use the Commercial Edition.

Type 'o' if you want to use the Open Source Edition.

键入:o

Type '3' to view the GNU General Public License version 3.

Type 'L' to view the Lesser GNU General Public License version

Type 'yes' to accept this license offer.

Type 'no' to decline this license offer.

Do you accept the terms of either license?

键入:yes

**1.3 编译源码**

执行命令:make -j4。加上-j4 选项表示开了 4 个线程进行编译,时间会大大缩短。

**1.4 安装程序**

Ubuntu 环境下命令为:sudo make install

安装成功后系统会生成这个文件: /usr/local/Trolltech/Qt-4.7.3/bin/qmake

**1.5 配置 QT 的环境变量**

ubuntu 环境下:

vim /etc/profile ，加入：

export QTDIR=/usr/local/Trolltech/Qt-4.7.3

export PATH=$QTDIR/bin:$PATH

export LD\_LIBRARY\_PATH=$QTDIR/lib:/usr/local/lib:$LD\_LIBRARY\_PATH

保存之后 source /etc/profile立即生效.

**1.6 测试**

输入命令:qmake –v

若输出以下信息表示 QT 安装并配置成功。

liugang@ubuntu:~$ qmake -v

QMake version 2.01a

Using Qt version 4.7.3 in /usr/local/Trolltech/Qt-4.7.3/lib

**1.7 安装 QCreator**

至此,qt-x86 环境已搭建好,但是这样编写代码开发 GUI 应用程序不方面,我们通过Qt Creator 可以进行快速开发。Qt Creator 是一个用于 Qt 开发的轻量级跨平台集成开发环境。

(1) 下载 QCreator

下载地址: http://download.qt.io/official\_releases/qtcreator/2.5/2.5.2/

下载版本:qt-creator-linux-x86-opensource-2.5.2.bin

(2) 安装:

执行命令:./ qt-creator-linux-x86-opensource-2.5.2.bin

根据系统提示选择即可,默认 QCreator 的安装路径是:/opt/qtcreator-2.5.2

安装成功后,Linux 系统桌面会出现 QCreator 图标。

## 2、QT实现MP3 播放器

### 2.1在QT Creator创建MP3播放器的QT项目

(1)打开Qt Creator，点击文件-新建文件或工程，选择Qt Gui应用

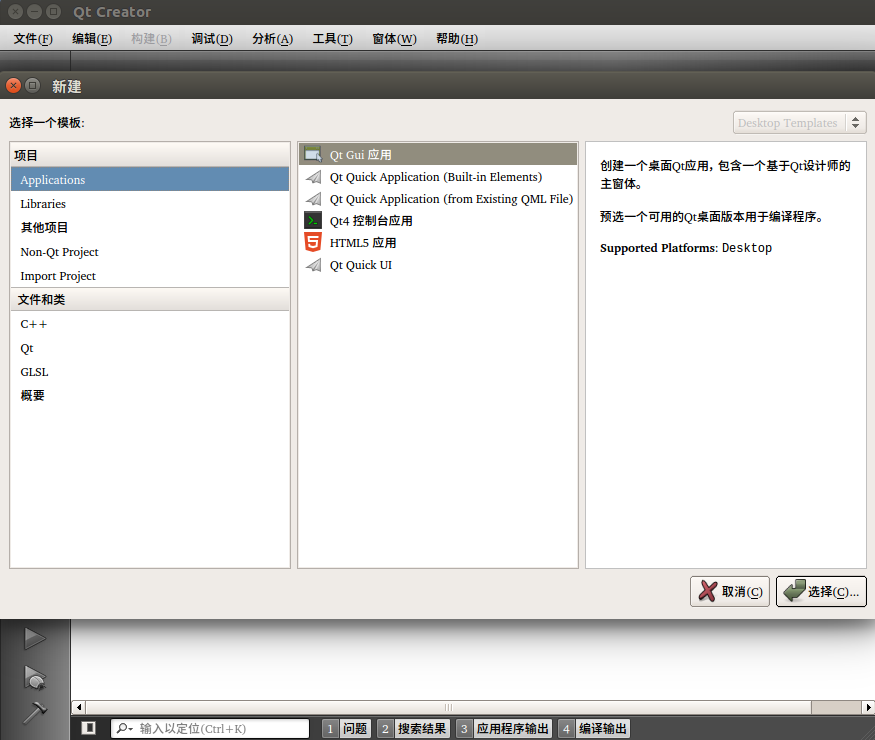


图2.1 1

(2)输入项目名称和项目创建路径

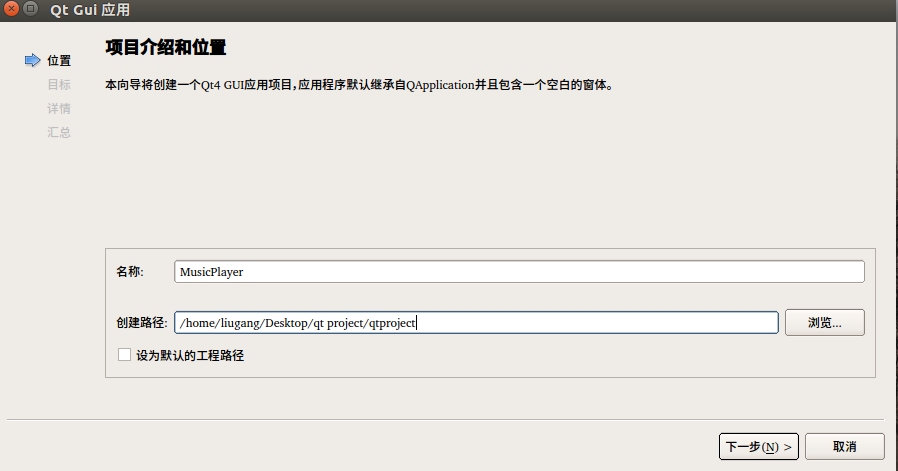


图2.1 2

(3)设置qt项目的基类为QWidget，并勾选创建界面

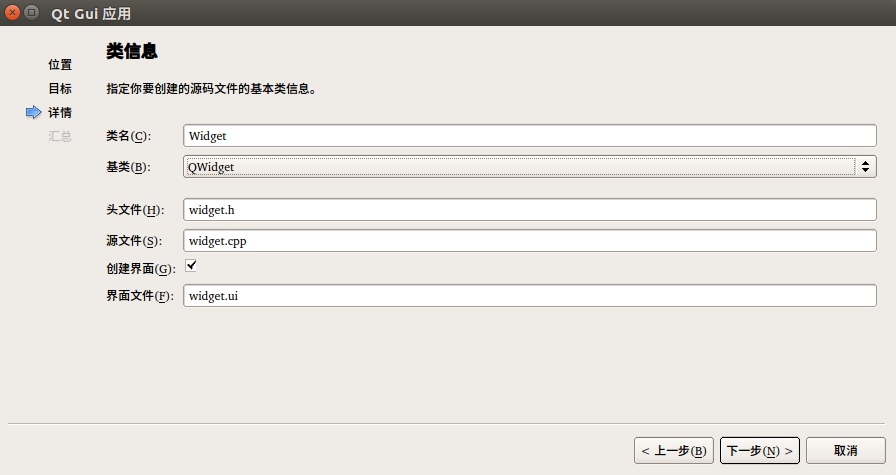


图2.1 3

(4)完成qt项目的创建

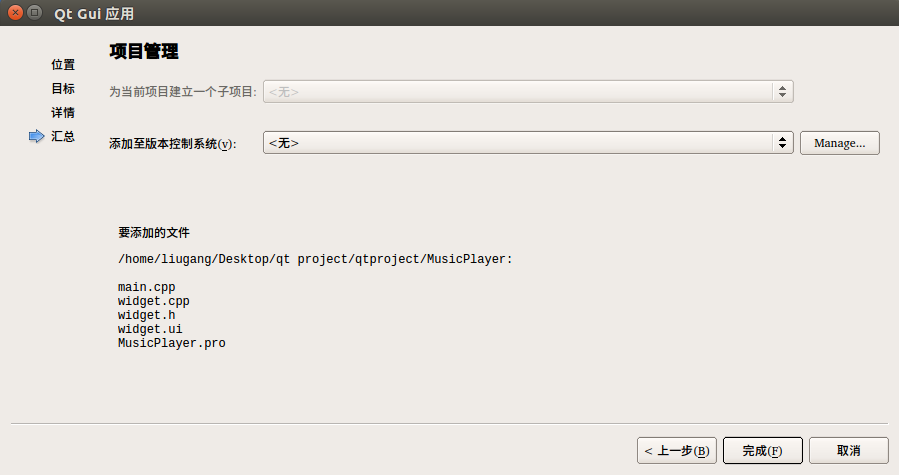


图2.1 4

### 2.2 MP3播放器的ui界面的设计

点击widget.ui进入ui界面，在界面的画板中添加若干label、push button、list widget、horizontal spacer（滑动条）等控件。设计的ui界面如下图：

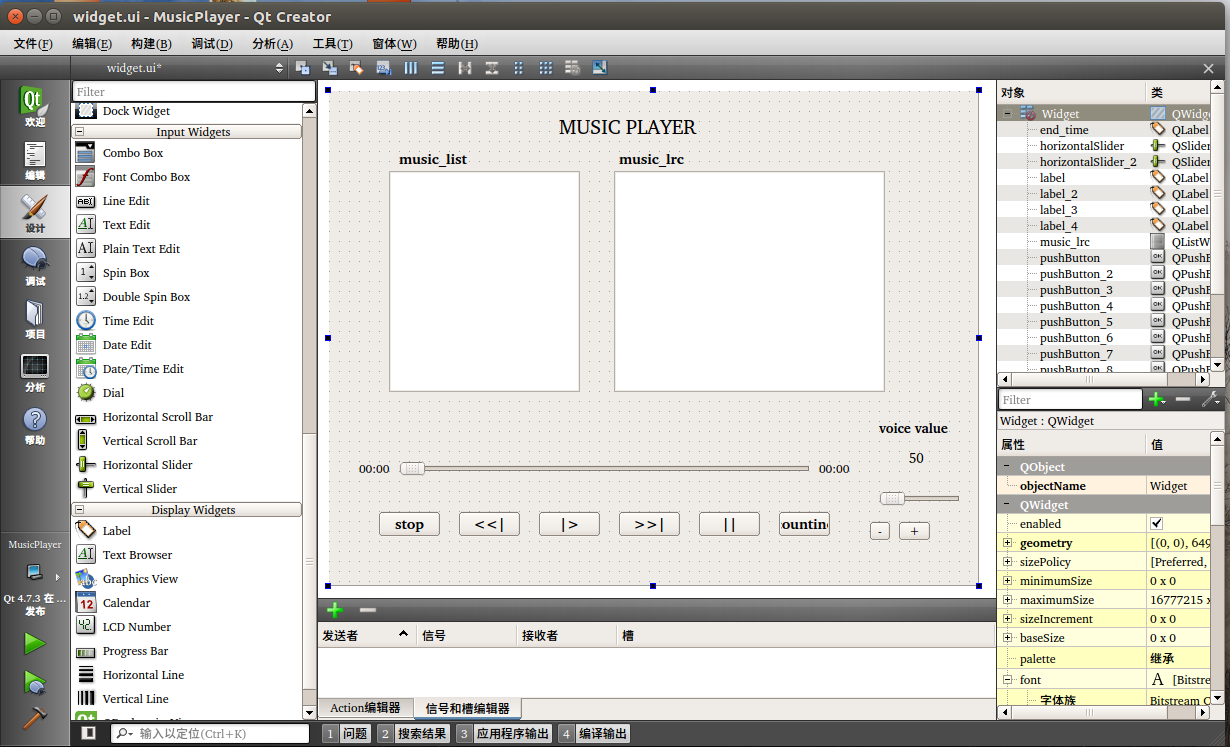


图2.2 1

### 2.3 UI相关控件槽函数的设计

2.3.1、槽函数的声明（widget.h）

private slots:

//---------单击播放按钮---------

void on\_pushButton\_3\_clicked();

//---------单击下一首按钮--------

void on\_pushButton\_4\_clicked();

//---------单击上一首按钮--------

void on\_pushButton\_2\_clicked();

//---------单击继续按钮---------

void on\_pushButton\_clicked();

//---------单击暂停按钮---------

void on\_pushButton\_5\_clicked();

//---------单击停止按钮---------

void on\_pushButton\_8\_clicked();

//---------双击歌曲--------------------

void on\_ui\_music\_list\_doubleClicked();

//---------声音进度条值改变------------------------

void on\_horizontalSlider\_2\_valueChanged(int value);

//---------单击声音-按钮---------

void on\_pushButton\_7\_clicked();

//---------单击声音+按钮---------

void on\_pushButton\_6\_clicked();

//---------定时器操作---

void start\_counting();

//---------歌曲进度条拖到释放---------------

void on\_horizontalSlider\_sliderReleased();

2.3.2、关键槽函数的实现

**2.3.2.1、实现歌单的展示**

首先调用系统命令生成歌曲清单存入文件中，之后打开文件进行一行一行地读取，将读取到的歌名添加到list widget中实现歌词的展示。

//----------------------------加载歌曲清单-----------------------

system("ls /home/liugang/Desktop/music/ > /home/liugang/Desktop/music\_list");

**fstream fin("/home/liugang/Desktop/music\_list");**

std::string buffer;

int music\_number=0;

while(getline(fin,buffer))

{

const char\* music=buffer.c\_str();

**ui->ui\_music\_list->addItem(tr(music));**

music\_number++;

}

**2.3.2.2、歌曲进度的时间、歌词动态显示和进度条的动态改变**

采用定时器来实现动态的操作，设置定时一秒，当超过一秒后，系统将产生一个timeout()信号，通过connect关联来执行相应的槽函数更改时间和进度条，从而实现进度条、歌词和时间的动态显示。

//----------------------------定时器连接函数--------------------- **QObject::connect(timer,SIGNAL(timeout()),this,SLOT(start\_counting()))**;

//-----------------------------定时器槽函数--------------------------

void Widget::start\_counting()

{

playtime++;

//--------------时间动态-----------

//caltime()函数将播放时间(int)转换成Qstring字符串

QString qplaytime=caltime(playtime);

**ui->start\_time->setText(qplaytime);**

//--------------进度条和歌词-----------------

**ui->horizontalSlider->setValue(playtime);**

int music\_row=ui->ui\_music\_list->currentRow();

if(playtime%3==0&&playtime>=15&&music\_row==0){

**ui->music\_lrc->setCurrentRow((playtime-16)/3);**

//此次歌词只做了一个简单的定位实现，因为时间和lrc文件的问题，此代码只作为测试使用，并不能实现歌词动态定位功能。

}

}

**2.3.2.3、单击播放按钮功能函数**

首先判断是否有歌曲播放，如果有，将播放的歌曲KILL掉。之后取出播放的歌曲在list widget中所在行的歌名来载入歌词，将歌词载入music\_lrc中。将歌名使用label进行显示，并使用madplay播放歌曲，开启定时器。

//------------------------------单击play按钮------------------------

void Widget::on\_pushButton\_3\_clicked()

{

if(!firstflag) system("killall -KILL madplay ");

int curMusicRow=ui->ui\_music\_list->currentRow();

**QString currentMusic=ui->ui\_music\_list->currentItem()->text();**

//---------读取载入歌词------------

fstream finj("/home/liugang/Desktop/lrc/JoelHanson-TravelingLight");

std::string buffer;

while(getline(finj,buffer))

{

const char\* musiclrc=buffer.c\_str();

**ui->music\_lrc->addItem(tr(musiclrc));**

}

//------------lable显示歌名--------------

**ui->show1->setText(currentMusic)**;

string com="madplay /home/liugang/Desktop/music/"+currentMusic.toStdString()+" < /dev/null > /dev/null 2>1&";

const char\* command=com.c\_str();

system(command);

firstflag=false;

//-------------开启定时器，定时1秒------------

**timer->start(1000);**

**2.3.2.4、音量调节进度条函数**

如果进度条的值改变，将会触发这个槽函数。首先将进度条的值通过label进行显示，通过调用系统命令amixer来改变音量。

//-------------------------------音量活动slider---------------------

void Widget::on\_horizontalSlider\_2\_valueChanged(int value)

{

QString voice\_value=QString("%1").arg(value);//int 转Qstring

**ui->voice->setText(voice\_value);** //设置label声音的值

string svoice\_value=intToString(value);

**string linux\_voice\_com="amixer -q sset PCM "+svoice\_value+"%";**

const char\* linux\_voice\_command=linux\_voice\_com.c\_str();

system(linux\_voice\_command);

}

**2.3.2.5、通过歌曲进度条来调节歌曲播放进度**

首先得到进度条的值，将这个int值转换为时间的表示方式。通过madplay的 -s 参数加入时间来控制歌曲播放进度。

//------------------------------歌曲进度条---------------------------

void Widget::on\_horizontalSlider\_sliderReleased()

{

system("killall -KILL madplay ");

**QString currentMusic=ui->ui\_music\_list->currentItem()->text();**

**int slidertime=ui->horizontalSlider->value();**

playtime=slidertime;

QString qplaytime=caltime(playtime);

**string com="madplay /home/liugang/Desktop/music/"+currentMusic.toStdString()+" -s "+qplaytime.toStdString()+" < /dev/null > /dev/null 2>1&";**

const char\* command=com.c\_str();

system(command);

timer->start();

}

其他槽函数的实现请参考所提交的qt代码文件。

### 2.4 UI界面的美化

2.4.1、编写qss代码并加入相应的资源文件

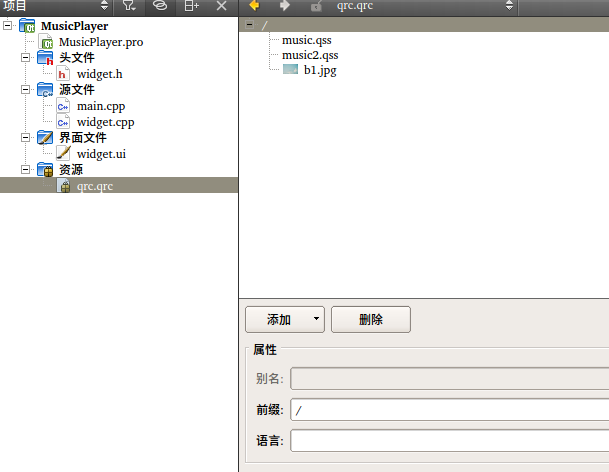


图2.4 1

2.4.2、在main函数载入qss文件

//------------加载qss-------------

QFile qss(":/music2.qss");

qss.open(QFile::ReadOnly);

qApp->setStyleSheet(qss.readAll());

qss.close();

### 2.5成员变量和头文件说明

2.5.1、成员变量

public:

explicit Widget(QWidget \*parent = 0);

int playtime; //播放时间

int playrow; //当前播放歌曲所处歌单的行数

bool firstflag; //播放标志位

QTimer \*timer=new QTimer(); //定时器

2.5.2、头文件信息

#include <QWidget>

#include <QTimer>

#include <QApplication>

#include <QFile>

#include <sys/types.h>

#include <fstream>

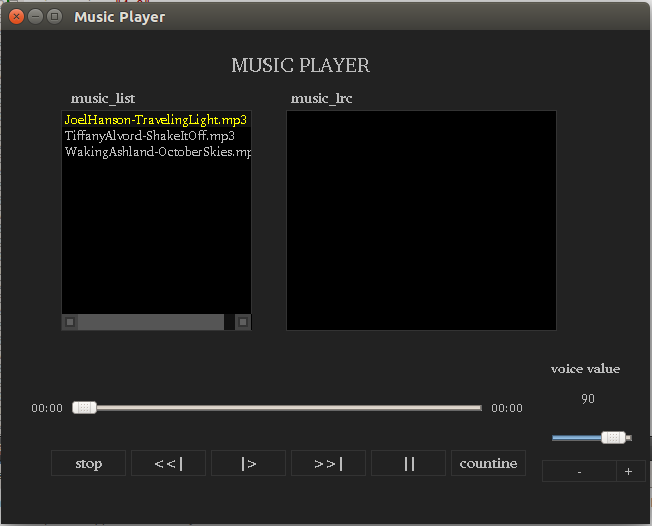
#include <QString>

#include "sstream

# 四、【运行结果】

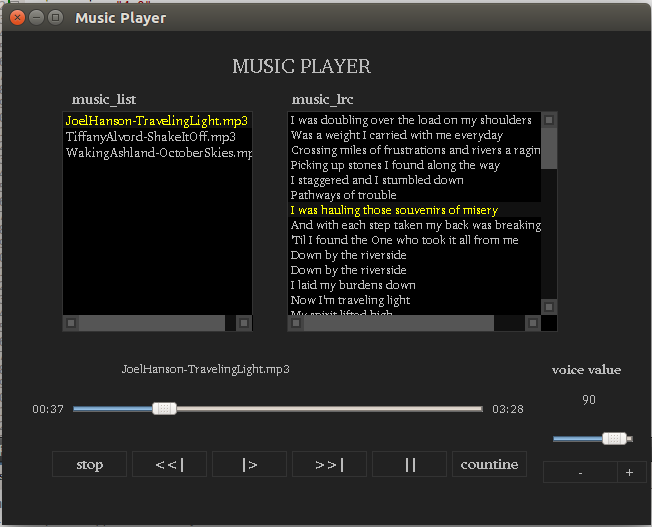
## 1、Qt MP3播放器的运行

1.1 初始界面



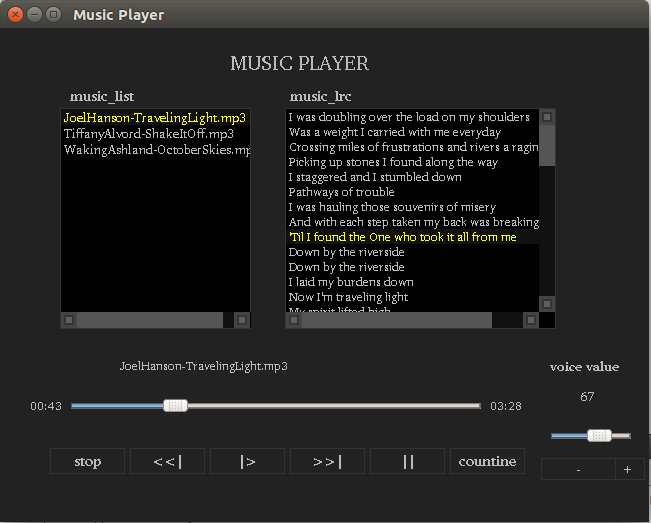
四 图1 1

1.2 点击播放，实现歌词、播放时间，播放进度条的动态显示



四 图1 2

1.3 使用进度条或者按钮实现声音的调节



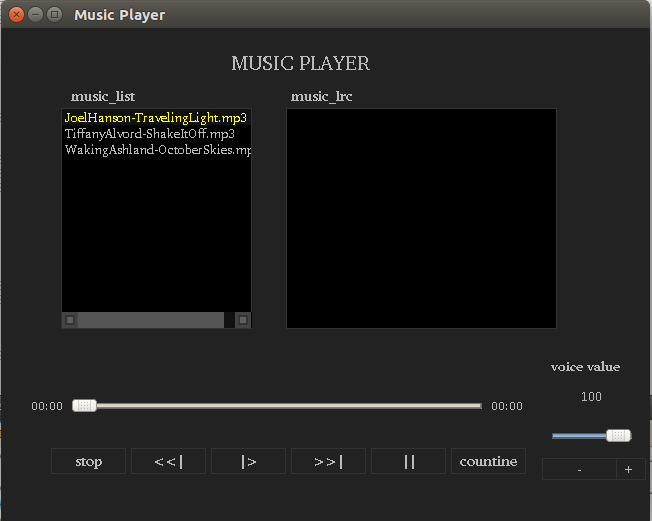
四 图1 3

1.4 单击上/下一首播放歌曲，实现暂停继续等



四 图1 4

1.5 点击stop停止歌曲播放



四 图1 5

# 五、【心得体会】

问题：在使用madplay时，使用system(“madplay 1.mp3 &”)不能正确地后台播放

解决办法：使用system(" madplay 1.mp3 < /dev/null > /dev/null 2>1&")进行后台播放。

问题：在设计MP3播放器歌词显示时，无法动态地显示歌词

解决办法：

1、定义下载的LRC歌词的格式如下：[00:11.36] 长亭外，古道边

2、获取当前的歌曲名称 ui->ui\_music\_list->currentItem()->text()

3、根据歌曲名称找到相应的LRC歌词，并将时间读取到lrc\_time[]数组，将歌词读取到lrc\_words[]数组。

4、通过定时器的槽函数来判断当前的时间应该处于歌词的哪行

for(int i=0;i<N;i++)

if(playtime<lrc\_time[i]) //如果当前播放时间小于这一行开始时间

{ ui-> ui\_music\_list->setCurrentRow(i-1);//定位上一行

break;//完成歌词的定位，退出循环

}

# 六、【附录】

## 1、widget.cpp源代码

#include "widget.h"

#include "ui\_widget.h"

#include <sys/types.h>

#include <fstream>

#include <QString>

#include "sstream"

using namespace std;

Widget::Widget(QWidget \*parent) :

QWidget(parent),

ui(new Ui::Widget)

{

**//----------------------------初始化ui----------------------------**

ui->setupUi(this);

this->setWindowTitle(tr("Music Player"));

playrow=0;

playtime=0;

firstflag=true;

**//----------------------------加载歌曲清单-------------------------**

system("ls /home/liugang/Desktop/music/ > /home/liugang/Desktop/music\_list");

fstream fin("/home/liugang/Desktop/music\_list");

std::string buffer;

int music\_number=0;

while(getline(fin,buffer))

{

const char\* music=buffer.c\_str();

ui->ui\_music\_list->addItem(tr(music));

music\_number++;

}

**//----------------------------初始化进度条--------------------------**

ui->ui\_music\_list->setCurrentRow(0);

ui->horizontalSlider\_2->setMinimum(0);

ui->horizontalSlider\_2->setMaximum(100);

ui->horizontalSlider\_2->setValue(90);

system("amixer -q sset PCM 90%");

ui->horizontalSlider->setMinimum(0);

ui->horizontalSlider->setMaximum(180);

**//----------------------------定时器连接函数------------------------**

QObject::connect(timer,SIGNAL(timeout()),this,SLOT(start\_counting()));

}

**//---------------------------ui析构函数-----------------------------**

Widget::~Widget()

{

delete ui;

system("killall -KILL madplay");

}

**//--------------------------播放时间转换成字符串--------------------**

QString caltime(int playtime)

{

QString qplaytime=QString("%1").arg(playtime);

if(playtime<60)

{

if(playtime<10)

qplaytime="00:0"+qplaytime;

else

qplaytime="00:"+qplaytime;

}

else

{

QString secondtime=QString("%1").arg(playtime%60);

QString minutime=QString("%1").arg(playtime/60);

QString second;

QString minu;

if((playtime%60)<10)

{

second=":0"+secondtime;

}

else second=":"+secondtime;

if((playtime/60)<10)

{

minu="0"+minutime;

}

else minu=minutime;

qplaytime=minu+second;

}

return qplaytime;

}

**//-----------------int to string -----------------------**

string intToString(int i)

{

stringstream ss;

ss << i;

string s1 = ss.str();

return s1;

}

//---------------slots------------

**//-----------------------------定时器槽函数-------------------------**

void Widget::start\_counting()

{

playtime++;

**//--------------start-time-----------**

QString qplaytime=caltime(playtime);

ui->start\_time->setText(qplaytime);

**//--------------horizontalSlider-----------------**

ui->horizontalSlider->setValue(playtime);

int music\_row=ui->ui\_music\_list->currentRow();

if(playtime%3==0&&playtime>=15&&music\_row==0){

ui->music\_lrc->setCurrentRow((playtime-16)/3);

}

}

**//------------------------------单击play按钮-----------------------**

void Widget::on\_pushButton\_3\_clicked()

{

if(!firstflag) system("killall -KILL madplay ");

int curMusicRow=ui->ui\_music\_list->currentRow();

if(curMusicRow==0)

{

ui->end\_time->setText("03:28");

fstream finj("/home/liugang/Desktop/lrc/JoelHanson-TravelingLight");

std::string buffer;

while(getline(finj,buffer))

{

const char\* musiclrc=buffer.c\_str();

ui->music\_lrc->addItem(tr(musiclrc));

}

}

if(curMusicRow==1)

{

ui->end\_time->setText("03:44");

ui->music\_lrc->deleteLater();

}

if(curMusicRow==2)

{

ui->end\_time->setText("04:18");

ui->music\_lrc->deleteLater();

}

QString currentMusic=ui->ui\_music\_list->currentItem()->text();

ui->show1->setText(currentMusic);

string com="madplay /home/liugang/Desktop/music/"+currentMusic.toStdString()+" < /dev/null > /dev/null 2>1&";

const char\* command=com.c\_str();

system(command);

firstflag=false;

timer->start(1000);

}

**//----------------------------单击下一首按钮------------------------**

void Widget::on\_pushButton\_4\_clicked()

{

system("killall -KILL madplay");

int currentMusicRow=ui->ui\_music\_list->currentRow();

int nextMusicRow=++currentMusicRow;

if(nextMusicRow>=ui->ui\_music\_list->count()) nextMusicRow=0;

ui->ui\_music\_list->setCurrentRow(nextMusicRow);

QString currentMusic=ui->ui\_music\_list->currentItem()->text();

ui->show1->setText(currentMusic);

string com="madplay /home/liugang/Desktop/music/"+currentMusic.toStdString()+" < /dev/null > /dev/null 2>1&";

const char\* command=com.c\_str();

system(command);

int curMusicRow=ui->ui\_music\_list->currentRow();

if(curMusicRow==0) ui->end\_time->setText("03:28");

if(curMusicRow==1) ui->end\_time->setText("03:44");

if(curMusicRow==2) ui->end\_time->setText("04:18");

playtime=0;

timer->start();

}

**//----------------------------单击上一首按钮------------------------**

void Widget::on\_pushButton\_2\_clicked()

{

system("killall -KILL madplay");

int currentMusicRow=ui->ui\_music\_list->currentRow();

int preMusicRow=--currentMusicRow;

if(preMusicRow<0) preMusicRow=ui->ui\_music\_list->count()-1;

ui->ui\_music\_list->setCurrentRow(preMusicRow);

QString currentMusic=ui->ui\_music\_list->currentItem()->text();

ui->show1->setText(currentMusic);

string com="madplay /home/liugang/Desktop/music/"+currentMusic.toStdString()+" < /dev/null > /dev/null 2>1&";

const char\* command=com.c\_str();

system(command);

int curMusicRow=ui->ui\_music\_list->currentRow();

if(curMusicRow==0) ui->end\_time->setText("03:28");

if(curMusicRow==1) ui->end\_time->setText("03:44");

if(curMusicRow==2) ui->end\_time->setText("04:18");

playtime=0;

timer->start();

}

**//----------------------------单击countine按钮---------------------**

void Widget::on\_pushButton\_clicked()

{

system("killall -CONT madplay");

timer->start();

}

**//------------------------------单击暂停按钮------------------------**

void Widget::on\_pushButton\_5\_clicked()

{

system("killall -STOP madplay");

timer->stop();

}

**//------------------------------单击stop按钮-----------------------**

void Widget::on\_pushButton\_8\_clicked()

{

system("killall -KILL madplay");

playtime=0;

timer->stop();

ui->start\_time->setText("00:00");

ui->end\_time->setText("00:00");

ui->show1->setText(" ");

ui->horizontalSlider->setValue(0);

firstflag=true;

}

**//-------------------------------歌曲清单双击播放-------------------**

void Widget::on\_ui\_music\_list\_doubleClicked()

{

system("killall -KILL madplay ");

QString currentMusic=ui->ui\_music\_list->currentItem()->text();

ui->show1->setText(currentMusic);

string com="madplay /home/liugang/Desktop/music/"+currentMusic.toStdString()+" < /dev/null > /dev/null 2>1&";

const char\* command=com.c\_str();

system(command);

playtime=0;

timer->start(1000);

}

**//-------------------------------音量活动slider--------------------**

void Widget::on\_horizontalSlider\_2\_valueChanged(int value)

{

QString voice\_value=QString("%1").arg(value);

ui->voice->setText(voice\_value);

string svoice\_value=intToString(value);

string linux\_voice\_com="amixer -q sset PCM "+svoice\_value+"%";

const char\* linux\_voice\_command=linux\_voice\_com.c\_str();

system(linux\_voice\_command);

}

**//------------------------------单击音量 - 按钮---------------------**

void Widget::on\_pushButton\_7\_clicked()

{

QString current\_voice=ui->voice->text();

int sub\_voice=current\_voice.toInt()-1;

ui->horizontalSlider\_2->setValue(sub\_voice);

QString voice\_value=QString("%1").arg(sub\_voice);

ui->voice->setText(voice\_value);

string svoice\_value=intToString(sub\_voice);

string linux\_voice\_com="amixer -q sset PCM "+svoice\_value+"%";

const char\* linux\_voice\_command=linux\_voice\_com.c\_str();

system(linux\_voice\_command);

}

**//------------------------------单击音量 + 按钮---------------------**

void Widget::on\_pushButton\_6\_clicked()

{

QString current\_voice=ui->voice->text();

int add\_voice=current\_voice.toInt()+1;

ui->horizontalSlider\_2->setValue(add\_voice);

QString voice\_value=QString("%1").arg(add\_voice);

ui->voice->setText(voice\_value);

string svoice\_value=intToString(add\_voice);

string linux\_voice\_com="amixer -q sset PCM "+svoice\_value+"%";

const char\* linux\_voice\_command=linux\_voice\_com.c\_str();

system(linux\_voice\_command);

}

**//-----------------------------歌曲进度条---------------------------**

void Widget::on\_horizontalSlider\_sliderReleased()

{

system("killall -KILL madplay ");

QString currentMusic=ui->ui\_music\_list->currentItem()->text();

int slidertime=ui->horizontalSlider->value();

playtime=slidertime;

QString qplaytime=caltime(playtime);

string com="madplay /home/liugang/Desktop/music/"+currentMusic.toStdString()+" -s "+qplaytime.toStdString()+" < /dev/null > /dev/null 2>1&";

const char\* command=com.c\_str();

system(command);

timer->start();

}

## 2、main.cpp源码

#include <QApplication>

#include <QFile>

#include "widget.h"

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

**//------------加载qss-------------**

QFile qss(":/music2.qss");

qss.open(QFile::ReadOnly);

qApp->setStyleSheet(qss.readAll());

qss.close();

Widget w;

w.show();

return a.exec();

}