班级 030731

学号 03073013

西安电子科技大学

本科毕业论文

题　　目 基于 Linux GUI 的视频播放器实现

学　　院 计算机学院

专　　业 教育技术学

学生姓名 朱春来

导师姓名 姚勇

毕业设计（论文）诚信声明书

本人声明：本人所提交的毕业论文《基于 Linux GUI 的视频播放器实现》是本人在指导老师指导下独立研究、写作的成果，论文中所引用他人的无论以何种方式发布的文字、研究成果，均在论文中加以说明；有关教师、同学和其他人员对本文的写作、修订提出过并为我在论文中加以采纳的意见、建议，均已在我的致谢辞中加以说明并深致谢意。

本论文和资料若有不实之处，本人承担一切相关责任。

论文作者：_____ 时间：____年____月____日

指导老师已阅：_____ 时间：____年____月____

西 安 电 子 科 技 大 学

毕业设计（论文）任务书

学生姓名 朱春来 学号 03073013 指导教师 姚勇 职称 讲师

学院 计算机学院 专业 教育技术学

题目名称 基于 Linux GUI 的视频播放器实现

任务与要求

本题目需要设计一个运行在 Fedora/Ubuntu Linux 系统下的多媒体视频播放器，通过 FFMPEG 和 MPLAYER 完成全格式媒体文件后台解码与播放，利用 Mono、GTK+或者 QT 完成界面的交互。要求：（1）熟悉 Linux C++（GTK+/QT）或者 C#（Mono）；（2）对 MPLAYER 有一定了解或对此有兴趣。

开始日期 2011 年 1 月 1 日 完成日期 2011 年 6 月

院长（签字）_____ 年 月 日

注：本任务书一式两份，一份交学院，一份学生自己保存。

西 安 电 子 科 技 大 学

毕业设计（论文）工作计划

学生姓名 朱春来 学 号 03073013

指导教师 姚勇 职 称 讲师

学 院 计算机学院 专 业 教育技术学

题目名称 基于 Linux GUI 的视频播放器实现

一、毕业设计（论文）进度

起 止 时 间	工 作 内 容
2011.1.15 —— 2011.3.15	查询阅读文档资料，对题目进行熟悉
2011.3.16 —— 2011.4.15	进行实践系统的概要、详细设计
2011.4.16 —— 2011.5.15	编码、调试并完善实践系统
2011.5.16 —— 2011.6.10	文献翻译以及撰写论文
2011.6.11 —— 2011.6.25	总结工作并完善论文

二、主要参考书目（资料）

1、《UNIX 环境高级编程》，作者： W.Richard Stevens，译者：尤晋元等
出版社：机械工业出版社，ISBN：711107579X，出版日期：2000 年 2 月
2、MPlayer & MEncoder, http://www.mplayerhq.hu/
3、《Foundations of GTK+ Development》，作者: Andrew Krause
出版社: Apress，出版日期: 2007-4-25

三、主要仪器设备及材料

PC 计算机，Windows 操作系统、Android SDK

四、教师的指导安排情况（场地安排、指导方式等）

实践场地安排于新校区，与指导教师保持在线沟通辅导，每两周进行一次见面指导与答疑，地点安排于新校区或本校区。

五、对计划的说明

无

注：本计划一式两份，一份交学院，一份学生自己保存（计划书双面打印）

摘要

Linux 有着良好的稳定性和伸缩性，其内核最小可以定制成 4KB 大小，特别适合作为嵌入式设备的操作系统，最新的 Android 操作系统就是采用 Linux 核心。

随着 Linux 在嵌入式方向的发展和桌面化效果越来越好后，人们希望在 Linux 环境下进行影音多媒体文件的播放。MPlayer 是 Linux 环境下最强大的视频播放软件，但是 MPlayer 在本质上是一个命令行程序，没有图形用户接口（GUI），因此需要为 Linux 开发一个有图形用户接口的视频播放器。

Mplayer 支持管道模式和后台模式；在管道模式下，可以通过管道文件播放视频文件，而在后台模式下可以通过管道文件来传递控制信号，从而达到控制 MPlayer 的效果。将 MPlayer 嵌入 GUI 中，一个基本功能具备的视频播放器就实现了。

因此，通过 MPlayer 作后台设备，PyQt 设计前台界面，可以快速开发在 Linux 中的基于 GUI 的视频播放器的实现 UPlayer。

关键词：视频播放器　**Linux　MPlayer PyQt　QT**

ABSTRACT

Linux has a good stability and scalability, and its core can be customized to the smallest size of 4KB, especially well suited for embedded devices operating systems, the latest Android operating system is the adoption of Linux kernel.

With the development of Linux in the embedded and desktop of the direction of getting better results, people want to playback multimedia file in the Linux environment. Mplayer is the most powerful video player in Linux environment, but MPlayer is essentially a command line program, there is no graphical user interface (GUI), so it's need to develop a Linux video player with graphical user interface.

Mplayer supporting FIFO pipeline mode and slave background mode.In the FIFO pipeline mode, the file can play video files through a pipe, while in the slave background mode to transfer files through the pipeline control signals, thereby to control the effect of MPlayer. Embedded in the MPlayer GUI, a basic function of the video player to have achieved

Therefore, by MPlayer for background equipment, PyQt interface design front, you can quickly develop in Linux, GUI-based implementation of the video player UPlayer.

Keywords: VideoPlayer Linux Mplayer PyQt QT

目录

第一章 绪论

1.1 Linux 中视频播放器概述

Linux 操作系统是广泛使用的自由软件，它具有开源、高效、稳定的特点。Linux 具有良好的可伸缩特性，Linux 内核最小可裁剪到 4KB，使的 Linux 在嵌入式设备方面有独特的优势。

普通 PC 用户和嵌入式设备用户迫切需要 Linux 对多媒体影音文件播放的支持，目前 Linux 中的常见视频播放器有 xine、Mplayer、Realplayer、KMPlayer 等。在这些现有的自由软件中，xine 严格意义上来说不是播放器，而是一个解码后台，主要负责视频解码工作；Realplayer 播放器的特点是支持流媒体拖放，可以作为浏览器的插件使用，但缺点是格式支持较少，主要用于 rm 格式视频文件播放。Mplayer 播放器的主要特点是功能强大，支持的格式种类多，且界面皮肤可以按用户喜好改变，和 xine 相比对中文支持更好，这对中文用户有很大的吸引力；而 KMplayer 播放器严格来说只是 Mplayer 的一个 GUI 实现。目前，Mplayer 是 Linux 中最强大的视频播放器。

1.2 基于 Linux GUI 的视频播放器

虽然 Mplayer 被称为 Linux 中最强大的播放器，但是，Mplayer 在严格意义上来说是一个命令行程序，需要一系列的开关选项来控制 Mplayer 的播放，并且 Mplayer 没有用户界面，是在类似 DOS 的终端或 shell 播放，除非是程序员等专业人士，一般用户很难很好的使用它。因此，在 Linux 中对 Mplayer 播放器实现 GUI 势在必行。

Linux 是“UNIX Like“的操作系统，不但继承了 UNIX 的设计思想，也继承了 UNIX 哲学。UNIX 哲学之一“只做一件事并把它做好”[8]，因此 MPlayer 的设计目标为设计一个视频播放器，而没有去关心用户界面，那样就不是只做一件事

了。UNIX 哲学之二“没有必要去重复发明一个轮子”[8]，因此我没有必要去重复设计一个视频解码器，而是借用 Mplayer 这个“轮子“，并只做界面设计这一件事情。所以，我可以借用 Mplayer 的视频解码功能做后台，而我自己只需要设计一个自己喜好的界面即可，我把这个项目叫 UPlayer，它的主要目标是为嵌入式设备实现一个基于 Linux 的 GUI 视频播放器。

通过 UPlayer 项目可以设计一个界面简洁、功能强大的视频播放器，自己设计播放器 GUI 的多个优点:

1.可以使用简单的界面元素控制 MPlayer，提供强大的功能

2.不用通过 MPlayer 源码重新进行 CMMI 过程编译，减少工作量

3.实现一个最简单的 GUI，把任务最简单化。

1.3 本文的研究成果和内容安排

1.3.1 本文内容安排

设计 UPlayer，工作主要分为菜单栏的设置、画面的嵌入、控制栏的绑定、侧边栏的设置、播放文件的更新几个方面，我把不同的工作分成不同的模块，方便管理和调试，同时，也有利于用户自定义功能或进行二次开发。

通过-wid 选项，在 PyQT 设计的 GUI 中选择一个 frame 来嵌入画面；通过 QProcess 类播放外部程序 Mplayer，并且通过 QProcess 进程类的方法 write（）来向 mplayer 传递 slave 模式控制命令，以达到控制 MPlayer 的效果[2]。

整个过程大致就如图 1 所示：

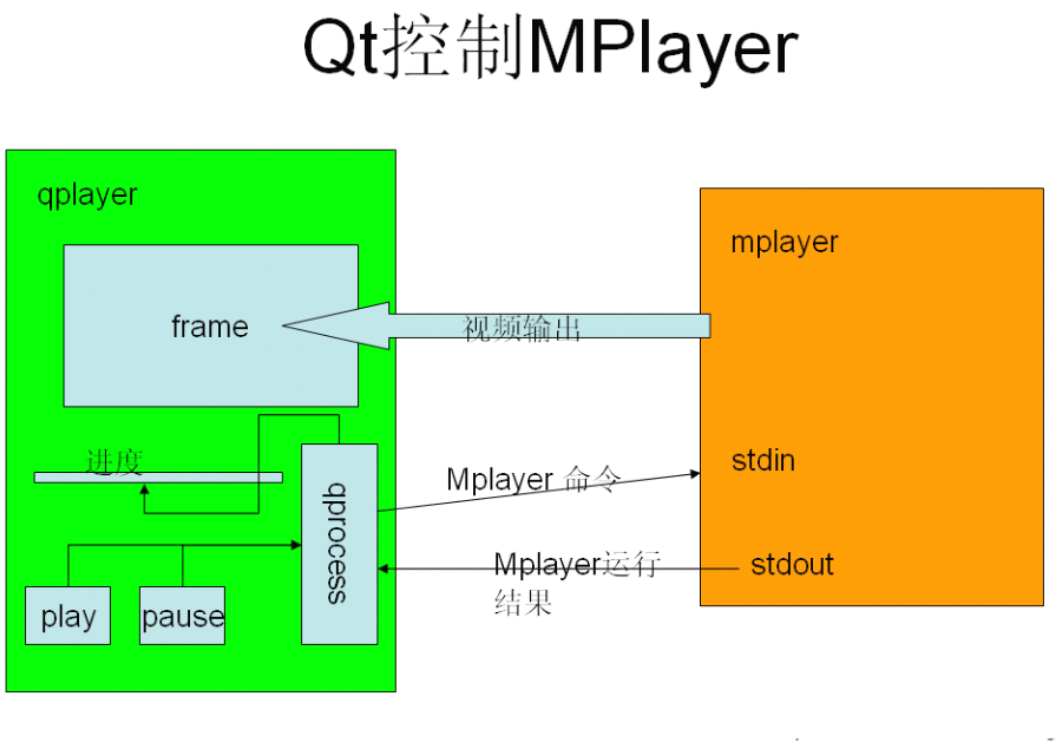

图 1 UPlayer 设计过程简图

Uplayer 程序一共设计四个模块类，把不同的工作放到不同的类，这样便于管理和组织，分别是 SettingMenuBar、SettingMovieScreen、SettingIcons 以及主类 UPlayer。其中 SettingMenuBar 类设置菜单栏工作，SettingMovieScreen 类负责画面嵌入和对画面的控制；SettingIcons 类是为了方便管理 UI 中使用的图标而专门设计的一个类。主类 UPlayer 通过继承这三个类来实现各项设置。

1.3.2 本文研究成果

经过一段时间的学习和反复实践，基本掌握了 PyQT 的 API 用法后，本着 GUI 尽可能简洁的原则，实现了 UPlayer 这个基于 Linux 的 GUI 视频播放器。

这是 UPlayer 整体效果图 2

图 2 Uplayer 的全貌

Uplayer 具有一般播放器的基本功能，比如全屏、静音、侧边栏、声音控制、进度控制等。UPlayer 的特点是 UI 设计十分的简洁，把常用的功能提供了 UI 元素，对于不是很常用的操作，比如截屏，则没有提供，以保持 UI 对资源的消耗最小；另外，对于 UI 的元素可以根据自己的喜好进行配置。不管是 GUI 还是 CLI，在 UPlayer 中都得到了很好的支持，而且 UI 设计的快捷键保持了和 MPlayer 本身的快捷键的统一性，比如增加声音是（*）、减小声音是（/）。简洁的 UI 设计，功能键的统一性以及操作方便是 UPlayer 的特点。

第二章 MPlayer 视频播放器

2.1 Mplayer 概述

Mplayer 软件是一个开源软件，具有强大的功能和很好的稳定性，是 Linux 中最强大的视频播放器，对于不完整的视频文件的处理更具有独到之处。另外，对于 Mplayer 还有各种播放模式，比如管道模式和 slave 后台模式，更多的选择是 Mplayer 具有更好的灵活性。

Mplayer 软件是 Linux 上最强大的视频播放器，它以 GNU 通用公共许可证（GNU Public License, GPL）发布。同时 Mplayer 具有跨平台特性，可以在主要平台使用，例如 GNU/Linux 以及其它类 UNIX 系统、微软公司的 Windows 系列系统以及苹果公司的 Mac OS X。因为 Mplayer 实际上是一个命令行程序，因此默认不带有 GUI，需要另外安装或自己编写 GUI。

2.2 管道模式

管道是一种很经典的进程之间的通信方式，其优点是简单易用，缺点在于功能简单。在 Linux 中管道相当于系统中的一个文件，来缓存所要传输的数据；但是当数据被读出后，数据就不存在于管道文件中了。

管道分为匿名半双工管道和 FIFO 管道，匿名半双工管道并不可以在文件系统中以任何方式看到。它只是进程的一种资源，会随着进程的结束而被系统清除，比如在 shell 命令中的“|”就是匿名半双工管道[15]。

FIFO 管道也叫有名管道，是一种文件类型，在文件系统中可以看到文件。在程序中可以通过查看文件 stat 结构中 st_mode 成员的值来判断文件是否是 FIFO 文件。在 FIFO 中可以很好的解决匿名半双工管道的缺点和限制，在无关联进程间进行数据的交换，表现出更加持久稳定的特性。因此，在 UPlayer 项目中，选择性能更好的 FIFO 文件来传递 MPlayer 的控制命令[15]。

一般的 I/O（open close read write unlink）函数都可以用于 FIFO 文件读写操作，FIFO 的出现极好的解决了系统在应用过程中产生的大量的中间临时文件的问题。FIFO 可以被 shell 调用把数据从一个进程到另一个进程，系统不必为该中间通道去烦恼清理不必要的垃圾数据。

当然 FIFO 管道也有局限性，在进程之间进行通信，进程发出请求前必须知道一个公共的 FIFO 通道，对于每个进程就要创建一个 FIFO 文件，如果进程很多，会造成系统过载。如果没有特别说明，在本文中管道就是指 FIFO 管道文件。

在 linux 中数据流重定向就是将某个命令执行后应该出现在屏幕上的数据，传输到其他地方，例如文件或设备（如打印机）。这在 CLI（命令行接口）中很重要，将本来在 CLI 输出信息重定向到 GUI（用户图像接口）中显示，是 UPlayer 中的习惯用法。

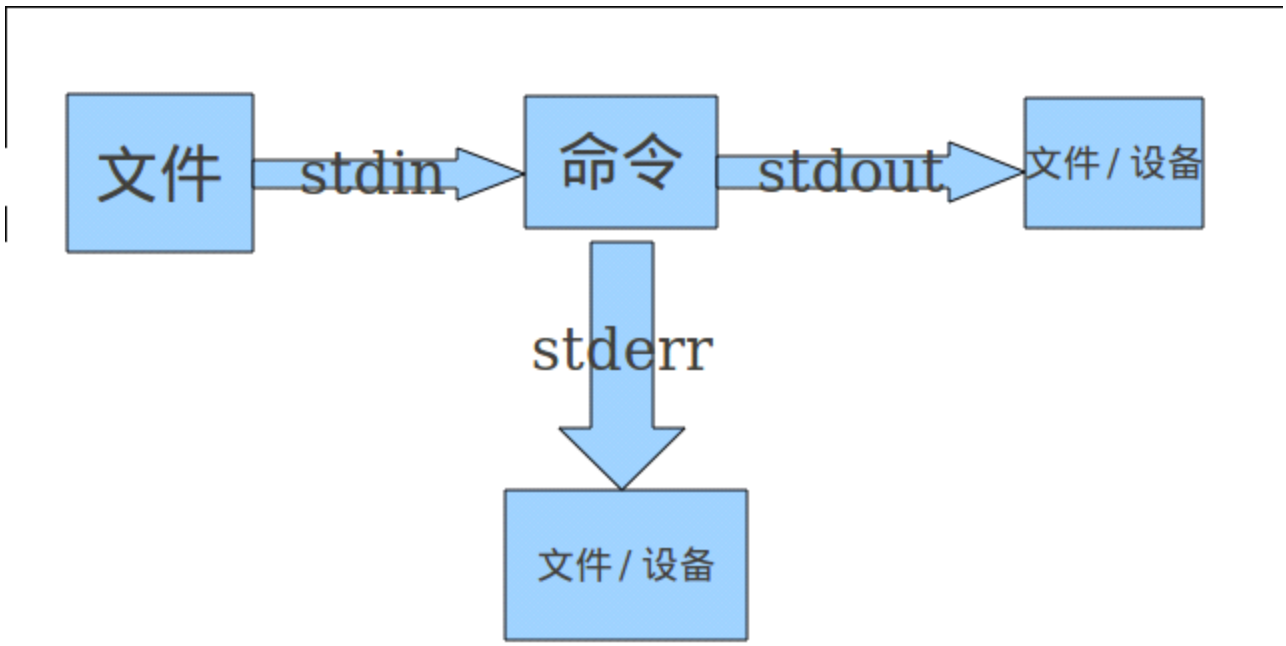

图 3 命令执行过程中的数据传输情况

在重定向过程中，可以将标准输出（stdout）与标准错误（stderr）传送到其他不同的地方，而不是在屏幕上，传送的目标通常是文件或设备（如图 3）[4]，传送命令如下：

1.标准输入（stdin），代码为 0，使用<或<<

2.标准输出（stdout），代码为 1，使用>或>>

3.标准错误输出（stderr），代码为 2，使用 2>或 2>>

2.3 Slave 后台模式

在 Linux 中“&”命令是的命令的执行在后台进行，而这里所指的后台模式是 MPlayer 的 slave 模式，要和“&”区别开来。在 slave 模式，MPlayer 为后台运行其他程序，仍然截获键盘事件，但 MPlayer 会从管道文件读一个换行符分隔开的命令。可以直接在 slave 模式下运行 mplayer，并通过 shell 来发送 slave 命令控制，但是这样的话就不方便在 UPlayer 的 GUI 中对 Mplayer 进行控制，因此采用管道文件结合 slave 模式的方法。

大多数 slave 模式命令相当于命令行选项，但并非一定要在相同的名称。常用的控制命令可以用“mplayer -input cmdlist”命令来查看，对于每个命令都要用行分隔符结束，并且新命令都会停止当前 Mplayer 的播放状态[1][2]。主要 slave 命令有控制播放进度、控制声音、控制播放状态、退出等。

2.4 指定播放窗口

MPlayer 是默认全屏播放（此处全屏指的是 Mplayer 充满窗口，而不是作为窗口的子窗口）,但在 UPlayer 中要指定区域或者控制来播放，这样方面用户操作，也更美观。MPlayer 通过提供窗口 Id 给参数-wid 来实现对播放画面指定播放窗口。

在对于正在运行图形界面窗口,可以用 Linux 命令 xwininfo 来取到其 ID,方法是运行这个程序后.屏幕鼠标变成+，将其拖到指定窗口即可看到其结果。Xwininfo 程序是 X-Windows 工具，用来查看已经打开窗口的 X-Windows 信息

[2]。对 UPlayer 的开发使用 PyQT 图像库，PyQT 的控件都是从 QWidget 类继承下来，因此它本身也有 wid,这样只要用 Qwidget.winId() 取出 WID，即可实现在指定窗口播放视频的功能。

第三章 QT 以及 PyQt 库

3.1 Qt 概述

Qt 工具箱是一个使用广泛的跨平台的 C++ 图形用户界面库，支持所有 Unix 系统，包括 Linux，还支持 WinNT/Win2k，Win95/98 平台和许多手持平台（如 Symbian）。Qt 具有良好结构化（但灵活）的面向对象的结构、清晰的文档以及直观的 API。基本上，Qt 同 X Window 上的 Motif，Openwin，GTK 等图形界 面库和 Windows 平台上的 MFC，OWL，VCL，ATL 是同类型的东西，但是 Qt 具有下列优点[19]:

1. 稳定性和跨平台性:

2. 封装良好的面向对象设计和开发

3. 提供了足够多的应用程序接口（API）

4. 支持 2D/3D 图形渲染，支持 OpenGL，

5. 大量的开发文档，QT 本身是开源的，拥有自己的社区

6. XML 支持

3.2 python 和 PyQT 概述

python 是一种面向对象的高阶解释脚本语言，包含了一组完善而容易理解的标准库，能够快速开发并完成常见的任务。在 linux 的 Ubuntu 版本下，默认安装了 python 解释器，通过命令 python 可以进入解释器，python 设计和开发哲学是“优雅、明确、简单“。

PyQt 是基于 Python 语言的 GUI 开发框架，不但是自由软件，还是是跨平台的；PyQt 可以运行于微软 Windows 系列、Mac OS X、GNU/Linux 以及 Unix 的多数变种上。实际上，PyQt 就是 python 化语法调用的 QT，可以非常方便的进行快速开发，又保持了 QT 的高效和稳定，对于对速度要求不是特别苛求的程序是个不错的选择[17][18]。

由于 PyQt 只是对 QT 采用了 python 的调用语法进行“克隆“的版本，本质上都是用 C++实现的，因此，在对没有特别说明的情况下，Qt 指的就是 Qt 和 PyQt 两者。

3.3 Qt 布局管理

Qt 中的布局管理包括水平布局、垂直布局和表格布局，可以用简图 4 表示：

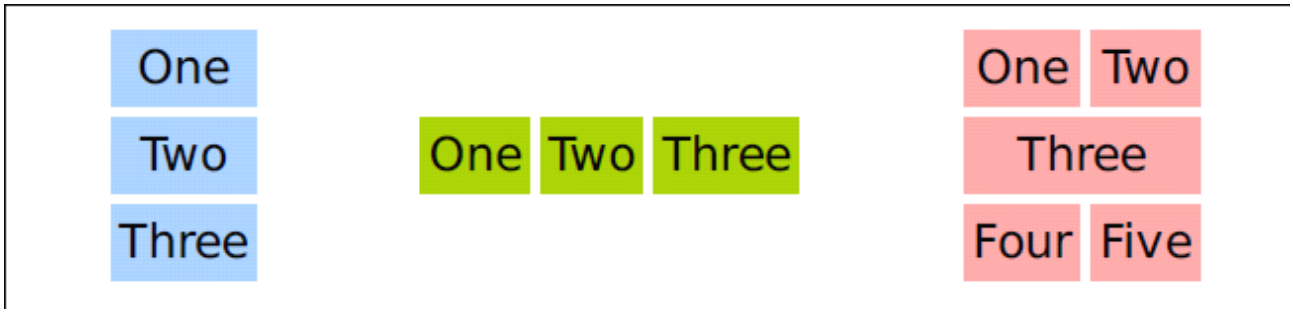

图 4 布局管理器

垂直布局如左边的排列，从上到下排列，使用布局管理器 QVBoxLayout；水平布局如图中间的排列，从左到右排列，使用布局管理器 QHBoxLayout；表格布局如图右边的排列，元素都填充在布局管理器的表格位置中，使用布局管理器 QGridLayout。

往布局管理器上添加元素采用函数 addWidget()，元素按添加顺序排列；添加分隔符是函数 addStretch(int)，int 参数决定分隔开的距离。在布局管理器中，有一个函数是 addLayout(QVBoxLayout|QHBoxLayout|QGridLayout），用来在布局管理器中添加另外的布局管理器，以达到良好的布局。对一个界面元素设置布局管理器是通过函数 setLayout()实现的，常用的方法是用布局管理器添加好元素，设置为主布局管理器[12][13]。

3.4 Qt 区域分布

Qt 将 GUI 分为一个中央区域和八个 Dock 区域，另外上下分别有 Menu Bar 区域、Toolbars 区域和 Status Bar 区域[12][18]，如图 5 所示：

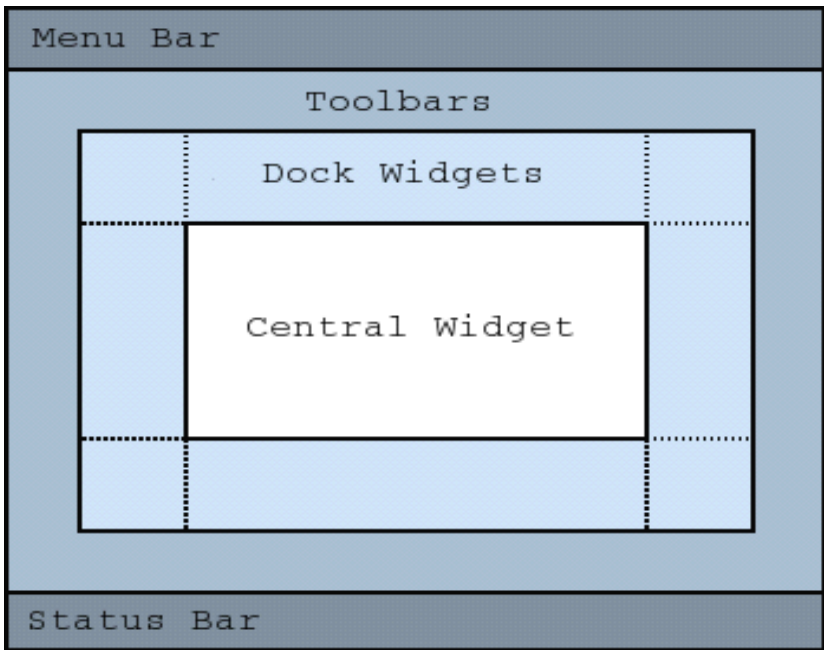

图 5 QT 分布区域

MenuBar、ToolBars 和 StatusBar 分别用于设置菜单栏、工具栏和状态栏。对于其它区域，如果没有设置的话，默认是关闭的，因此整个画面只有中央区域显示。 通过 QDockWidget()创建 Dock Widget，然后通过 setAllowedAreas()使得侧边栏在右侧显示；对于控制栏的添加则采用了另一种方法，就是直接添加，而不是通过控制显示区域实现。对于右下角（序号为 3）这个区域，设计为作为侧边栏的控制按钮，通过函数 setCorner(Corner(3),RightDockWidgetArea)实现这个区域被侧边栏占有。

3.5 信号和槽机制

3.5.1 信号槽概述

QT 的核心机制是信号和槽机制，信号和槽是一种接口，用于 QT 对象之间的

通信，是 QT 和 Tkinter、GTK+等工具包的主要区别。一般的工具包进行通信都是通过回调函数，而这种机制很不稳定，会产生大量的垃圾代码。QT 是由 C++语言定义的，且其语法也类似 C++语言，继承了 C++的高效和高度面向对象化设计和开发。

所有从累 QWidget 派生的类都有信号和槽，当对象的状态改变时会发送信号，而当信号发送出去后，由槽来处理信号。对象只负责发送信号，这样更具有封装性，也更高效。槽其实是一些普通的函数，通过信号来触发事件，信号和槽机制是 QT 的内部机制，对象并清楚信号如何发送到槽的，而槽也不知道信号如何发送过来的，但是槽只接受信号来启动事件处理函数。

对于 PyQT 来说，它把 QT 进行 python 化，集合了 python 的语言简洁和快速开发的能力，以及 QT 的高效稳定，对于大型程序，PyQt 的优势越明显[19]。

3.5.2 信号（SIGNALS）

信号是在对象状态改变时发送的，而只有对象定义了信号才能发送，对于没有定义的信号无法发送，默认忽略这种状态的改变。当一个信号被发送（emit）出去以后，能够捕捉这种信号的槽机制立即启动，对信号进行处理，类似 Tkinter 的回调函数。在 PyQT 中信号和槽是与 GUI 无关的，之间相互独立，是在 PyQT 内部进行，设计者不必关系具体的细节。如果当一个信号发送出去后，有至少一个槽捕捉到这个信号，都会对信号进行处理。比如声音大小改变后，会发送信号，而 Mplayer 会捕捉这个信号并进行声音改变，而声音进度条也会捕捉这个信号并改变值，但谁先谁后没有规定，这是随机过程[19][11][14]。

3.5.3 槽（SLOTS）

在 PyQT 中，槽就是函数，可以是系统定义或自定义函数，它的特点是可以通过接受信号来调用。槽关联的信号要自己设置了，除了设置关联的信后，对其他信号一律忽略不计。

槽是用 C++实现的，因此包括 public、private 和 protected 三种类型：

公共槽：在这个区内声明的槽意味着任何对象都可将信号与之相连接。

受保护槽：在这个区内声明的槽意味着当前类及其子类可以将信号与之相连接。

私有槽：在这个区内声明的槽意味着只有类自己可以将信号与之相连接[19][11]。

3.5.4 信号与槽的关联

通过调用 QObject 对象的 connect 函数来将某个对象的信号与另外一个对象的槽函数相关联，这样当发射者发射信号时，接收者的槽函数将被调用。

这个函数的作用就是将发射者对象中的信号与接收者的槽函数联系起来。当指定信号 signal 时必须使用 QT 的宏 SIGNAL()，当指定槽函数时必须使用宏 SLOT()。如果发射者与接收者属于同一个对象的话，那么在 connect 调用中接收者参数可以省略。

当信号与槽没有必要继续保持关联时，可以使用 disconnect 函数来断开连接。这个函数断开发射者中的信号与接收者中的槽函数之间的关联[19][13]。

使用 disconnect 函数的情况有：

第一、断开与某个对象相关联的任何对象。使用方法是对象直接调用这个函数，如 self.lable.disconnect()；第二、断开与某个特定信号的任何关联。如 self.label.disconnect(QtCore.SIGNAL("clicked()"))；第三、断开两个对象之间的关联。如 self.label.disconnect(self.button)

在 disconnect 函数中 0 可以用作一个通配符，分别表示任何信号、任何接收对象、接收对象中的任何槽函数。但是发射者不能为 0，其它三个参数的值可以等于 0。

3.5.5 注意的问题

信号和槽机制是十分稳定高效的机制，在大型程序中尤其如此，但是运用信号槽机制要注意这些问题[19]：

1.对执行效率要求苛刻的情况下慎用，因为信号发出后要有一个内部槽捕获信号的过程，会耗费一段时间；

2.不要在槽函数中发送信号，很容易产生死循环，槽函数设计只接受处理信号，信号的发送由其它对象进行；

3.不要试图改变槽捕获信号的次序，因为这在内部处理时是随机进行的。

4.不要给信号和槽设置缺省参数，虽然有时候这样很方便，但是，尽量让一

切条件由信号提供，而槽只负责信号的处理。

3.6 QProcess

在 QT 中使用 QProcess 这个类来执行外部程序[2]。在 PyQt 中通过 QProcess 创建一个进程类对象后，执行 start 方法启动外部程序。为了给外部程序传递信息，QProcess 类有 write 方法来向标准输入传递信息，达到控制 QProcess 执行的外部程序的目的。 通过 I/O 重定向，当有输出信息时，通过过滤分析函数进行分析过滤获得有用信息。

Qprocess 类有 canReadLine 函数来判断是否有信息输出用于分析，如果有信息的话，则用 readLine 函数获得信息，这个信息是 QByteArray 类型的，类似与字符数组。这里对信息的分析工作主要有：

1.分析播放文件的名字，用于添加到侧边栏；

2.分析播放文件的时间长度，用于设置播放进度条变化范围，setRange(0,int(self.length))

3.分析播放位置，用于设置状态栏信息更新

第四章 UI 设计

4.1 菜单栏和工具栏概述

一般的 GUI 程序都有菜单栏，对于菜单项比较多的情况，还会把主要功能专门设置一个工具栏，提供功能热键，比如 Word 程序。菜单栏主要设计包括文件、编辑、查看、转到、声音和帮助菜单项，工具栏对主要功能如打开文件、退出等功能设置了热键，下面对各个菜单项进行详细叙述。

运用一个 settingMenuBar 函数来作为设置菜单栏和工具栏的主函数，通过 self.menuBar()创建了菜单栏，通过 self.addToolBar(u”工具栏(&T)”创建了工具栏。在前面 QT 的区域介绍中曾介绍过，QMainWindow 是默认含有菜单栏、工具栏、状态栏等属性的。这里的 UPlayer 类继承 QMainWindow 类，默认也含有这些属性。虽然 SettingMenuBar 类没有这个属性，但是，UPlayer 类会继承这个类，也就等同于在 SettingMenuBar 类可以使用这些属性，如果是单独使用这个类的话，有可能要进行修改。创建菜单项是通过 self.menubar.addMenu(u”文件(&F)”)等创建了菜单“文件“等菜单。

4.1.1 文件菜单

一般文件菜单负责与播放文件有的操作，文件菜单主要包括“打开“、”打开位置文件“和“退出“项，其中“打开“和“打开位置文件“的区别是：“打开”是打开一个选择文件的对话框来选择播放文件，而“打开位置文件“是一个输入对话框来输入正确的文件目录地址，用户必须事先知道播放目录。

对于一个字符串前面的“u“是告诉解释器，这个字符串采用 Unicode 编码，提高 UPlayer 程序的可移植性，这对中文化有很大帮助。setShortcut(),设置快捷键为 Ctrl+O；open.setStatusTip(u”打开文件”),设置状态提示信息，但鼠标移到“打开“菜单项上，在状态栏会显示“打开文件“的提示信息，告知用户此菜单项的功能简要描述。addAction()是“文件“菜单增加一个菜单项“打开”。

self.connect(self.open,QtCore.SIGNAL(“triggered()”),self.showFile

_dialog)，这里运行了 PyQT 里面的信号和槽机制，但 self.open 菜单项被点击，会发送“triggeed()“信号，通过槽传递，引发函数 self.showFile_dialog 的执行。

通过 QFileDialog 类的 getOpenFileName(self,u”打开文件”,”/home”)函数获取文件名，其中“/home“是默认开始目录。获取的文件名是 filename 变量，如果没有获得文件名，则退出函数，否则，将文件名传递给函数 self.change_movie_file(self, filename = None)[12][13][6]。

QtGui.QFrame 类创建自己的对话框，对话框由一个可编辑的输入行以及“确定“、”取消“两个按钮组成。“确定”按钮绑定函数 open_newfile_dialog，“取消“按钮绑定了函数 open_no_dialog，“退出“菜单项的代码不同之处是“退出“菜单传递“triggered()“信号，UPlayer 接收到信号后采取“close()”反应。

4.1.2 编辑菜单

编辑菜单主要关心 UPlayer 的自定义操作，这里没有提供首选项设置功能，但是提供了播放模式和静音控制，由于在 4.2 小节“文件菜单“中对创建菜单项介绍过了，这一小节主要介绍如何通过管道命令来控制播放模式和静音模式。

“循环模式”命令是"set_property loop -1\n"，随机模式命令"set_property loop 0\n"。在 mplayer 的 Slave 模式下，命令“set_property“用于设置变量的值，随机模式就是设置变量 loop 的值为-1，循环模式设置 loop 变量为 0 。

在对于静音模式的实现过程中，经过两个阶段，第一个阶段我没有直接使用 slave 模式下的命令，设置静音"set_property volume 0\n"，取消静音"set_property volume 100\n" 。这里其实并没有实现“静音“功能，因为静音是这样一个过程：开启静音，声音为 0，并保护现场，取消静音，声音恢复到之前状态。第一个阶段只是简单的设置声音为 0 或最大化（100 为最大化），并不是严格意义上的静音控制。

第二个阶段的代码采用了 slave 下的 mute 静音命令功能实现："mute 0 /1\n“。当 mute 的值为 0 时，Mplayer 处于静音模式，当 mute 的值为 1 时，MPlayer

处于正常播放模式。[2]

4.1.3 查看菜单和全屏、显示控制

查看菜单主要负责关于视频画面的控制工作，主要是全屏控制和显示控制，对于“全屏“控制，按照 slave 命令是“set_property vo_fullscreen 1\n”，但是我经过多次测试都没有实现，有可能是命令出错或者是窗口设置问题。因此，UPlayer 实现全屏功能最后采取了嵌入窗口的全屏显示属性来实现，和 Mplayer 全屏显示在效果上是一样的。

这里面 self.control_bool 和 self.control_dialog()是下面要讲述的“显示控制“菜单项的内容，先讲解如何实现全屏显示。通过一个逻辑变量 fullscreen_bool 来判断 Mplayer 是否处于 FullScreen（全屏）状态，当在窗口模式下，由设置 showFullScreen()函数来达到全屏效果；在全屏显示状态下，由设置 showNormal()函数来达到窗口显示效果。

在用户想不用切换到全屏模式，却达到播放画面最简洁化的情况下，希望通过操作可以把操作控制按钮等界面元素隐藏起来，这就是“显示控制“菜单项的功能。”显示控制“的实现是通过一个逻辑变量和一个开关状态来达到目的的，和 Mplayer 全屏效果的实现方法类似。“控制栏“和“侧边栏“都有函数 close()、show()来控制隐藏、显示状态。

可以看出，隐藏控制显示是把侧边栏、控制栏和工具栏都隐藏，在显示控制信息的时候，只把控制栏显示出来，这是考虑到用户一般显示控制信息是为了控制视频播放，而不是对文件列表的侧边栏和工具栏进行控制。

其中，图形部件的显示和隐藏分别采用方法 show 和 close，其中 show 方法有多个版本，包括 showNormal 和 showFullscreen。这样的话，任何一个部分都可以实现全屏显示，对于要嵌入多个画面的程序，可以提供解决方案。

对于侧边栏的显示控制，由于在下文对侧边栏有专门的叙述，这里先不介绍。

4.2 转到菜单和播放控制

转到菜单实际上是播放进度控制菜单，在这一节将介绍 UPlayer 使用最多的功能，那就是对 Mplayer 播放的控制，简单的几个按钮设计，实现了“播放/暂

停“、”前进/倒退“，”快进/快退“等功能。

4.2.1 播放控制

播放控制主要通过 slave 模式命令 seek 来实现的，seek 命令：

seek <value> [type] type 分为 0，1，2

0 ——表示对 value 是在现在基础上+/-操作

1——表示对 value 是总体进度的 value%操作

2——表示对 value 是总体进度中的第 value 秒

4.2.2 时间进度条

对“时间进度条“的控制涉及两个方面，一是通过时间进度条控制 Mplayer 的播放进度，二是 Mplayer 播放进度的改变引起时间进度条进度位置的改变，下面我从这两方面进行阐述。

通过 QtGui.QSlider 才创建一个进度条对象，方法 setRange（int，int）可以设置进度条的变化范围，通过 slave 命令”get_time_length\n”获得视频文件长度 length，再调用 setRange(0,length)确定时间范围。每当 slider 的值改变是，可以通过方法 value()获取改变后的值，达到时间进度条控制 Mplayerde 目的。对于 Mplayer 控制时间进度条，则主要是通过命令”get_time_pos\n”来获取播放位置，调用方法 setValue(int)来设置时间进度条的值。对于通过按钮实现的“快进、快退“等操作，也通过同样的方法控制时间进度条。

4.3 声音菜菜单和声音控制

声音菜单主要工作是控制声音大小，实现声音增大和减小的 slave 命令是：

Self.process.write(“volume +1\n”)

self.process.write(“volume -1\n”)

对增大或减小声音的快捷键保证了与 MPlayer 的统一，习惯了 MPlayer 快捷键的用户可以直接使用 UPlayer 的快捷键，两者设置一致，给用户带来了方便。

因为 MPlayer 的声音变化范围是 0～100，因此通过 setRange(0,100)使得进度条的变化在 0 到 100。当值为 0 时，通过信号启动处理函数来控制 Mplayer 把声音设置为静音。当值为 100 时，设置 MPlayer 为最大声音。

通过 self.sound_slider 的函数 value()可以获得进度条的值，再通过 set_property value 2\n 来改变 Mplayer 声音 。而当 Mplayer 的声音通过各种方式改变时，相应的要改变进度条的值。

在 Mplayer 声音改变时，对进度条的值进行相应改变，以保持对声音控制的一致性。在这方面，比对播放进度的控制设计的更好，以为播放进度控制出现了不一致的情况，还在不断改进当中。

4.4 帮组菜单和自定义对话框提示信息

帮助菜单提供程序使用手册和本版信息，一般程序的 F1 快捷键绑定的都是该程序的帮助使用手册，UPlayer 秉承了这一优良传统，当用户使用 F1 快捷键时，会获得一个对话框形式的帮助文档。

这是一个非常 简单的手册，对于详细的帮助手册，正如此对话框中所说明，可以使用 man uplayer 命令获得，uplayer.1.gz 这个使用手册对 UPlayer 有详细的介绍。之所以设计为非常简单的使用手册，是因为 UPlayer 就是设计一个非常的 GUI，若要深究 MPlayer 的使用，就该查询 MPlayer 的使用手册，而不是 UPlayer 的。对于快捷键 Ctrl+A 会获得 UPlayer 的版本信息，提供 UPlayer 的授权和作者信息。

4.5 侧边栏

一般播放器都会设置侧边栏，用于存放播放列表，方便用户查看或重播放文件，在具有侧边栏的情况了，“重复模式“和”随机播放模式“才有效。对侧边栏的设计主要涉及文件的添加和删减，文件排序的改变等。

4.5.1 播放文件列表信息

对侧边栏存储的播放列表主要控制信息有“添加“、”删除“、”清空“，“上移“和“下移“。对于“添加“，PyQt 中 QFileDialog.getOpenFileNames 方法获得一个文件列表，直接可以存放到侧边栏。其他操作，在 PyQt 中有一个类 QListWidget 专门用来列出文件列表，有方法 addAction、addActions 、

addItem、addItems、addScrollBarWidget 等用于”添加“对象，方法 clear 用于”清空“，方法 clearFocus 用于删除，方法 nextInFocusChain 和 previousInFocusChain 分别用于“下移“和“上移“。

4.5.2 文件和视频更新

对于文件的更新以及视频画面的更新，主要涉及函数 chang_movie_file，在函数对更新文件后再次播放前进行了一系列的工作，保证了前一个文件的正常结束，并为新文件的播放进行了初始化，Self.movieFile 属性存放的是播放文件的路径，通过重新设置属性可以更新播放文件。通过 process.write(“quit\n”)可以保证正播放文件的正常结束。

函数 self.play_event_default()是设置控制栏的“播放/暂停“按钮的图标，以保持一致性，主要是涉及一个逻辑变量的改变。而函数 self.setCmd 是更新 Mplayer 的管道文件，并通过 self.slider.setValue（）设置播放进度为 0，这些工作都是为了保持一致性，在更新画面前的初始化工作。真正更新视频画面的是 self.update_movie()函数，这个函数在 QProcess 介绍部分介绍过。

第五章 发布

5.1 发布概述

对于 GNU/Linux 平台可以进行源码发布，但是这样对于非 python 程序员用户，可能 PyQt 模块的安装都会让人望而却步，因此可以采用工具包 Distutils。对于 Windows 用户，习惯双击程序安装后便可使用，可以使用拓展程序 py2exe，建立独立的 Windwos 可执行程序。

5.2 文档补充

5.2.1 需求说明

在测试之前必须有一份精确的需求说明，这样测试才有参考目标，否则怎么知道 UPlayer 是否通过测试。需求说明是描述程序必须满足的需求的文档或者一些简短的笔记。程序设计的理念就是以编写测试程序开始，然后编写可以通过测试的程序。测试程序就是 UPlayer 的需求说明，它帮助我在开发程序的时候不偏离需求。

其实，在 UPlayer 中，UPlayer 类就是对其他模块的测试用例。通过把菜单栏设置、图标元素设置、嵌入画面等功能分成各个模块，通过 UPlayer 类来调用，可以立即查看各个模块的效果，达到单元测试的效果。

5.2.2 man 文档补充

在 Linux 系统中，不论是 C 语言的头文件，还是系统的程序，都可以通过 man 命令才参考使用手册，对于 UPlayer 这样一个软件，提供必要的 man 文档是必须的。

创建可以通过 man 命令查看的使用手册，可以使用软件 txt2man，通过命令 txt2man -t uplayer -s 8 > uplayer.man 创建，其中-t 参数是查询的标题，这里是 uplayer，而-s 是标识 uplayer 命令的属性，标识 8 表示 uplayer 是一个自定义的命令[4][15]。

不同的文档类别编号对应关系如下：

1	用户级别的命令	如 ls，man
2	系统调用	如 gethostname
3	库函数调用	如 isupper，getchar
4	特殊文件	如 fd，fifo
5	配置文件	如 ldap.conf
6	游戏和文稿演示	
7	杂项	
8	根用户运行的命令	如 UPlayer
9	内核文档	

然后对 uplayer.man 进行压缩,命令 gzip uplayer.man > uplayer.1.gz，将 uplayer.1.gz 复制到/usr/share/man/man1/中即可用 man 命令查看 uplayer 使用手册了。

5.3 常规打包发布

5.3.1 源码发布

建立最简单的 Distutils 安装脚本（setup.py）：

```
from distutils.core import setup
setup(name='UPlayer',version='1.0',description='A Simple Mplayer GUI Program',author='zhubuntu',py_modules=['UPlayer'])
```

，其中，py_modules 是要包含的模块列表确保 UPlayer.py 在本目录中。

运行命令 python setup.py build，会创建叫做 build 的子目录，其中包含名为 lib 的子目录，并把 UPlayer.py 的一个副本放置在 build/lib 中，这样打包工作就完成了。如果要在用户机器上进行安装，只需运行命令 python setup.py install[6][7]。

5.3.2 Linux 版本打包

python setup.py sdist 命令建立存档文件（适合源码发布），之后会创建 MANIFEST 文件，包括所有文件的列表。

这时候，增加了 dist 子目录，里面有一个 gzip 格式的 tar 存档文件，可以用这个文件进行源码发布。对于发布文件类型格式，可以用—formats 进行设定，支持格式有 bztar（bzip2 格式的 tar 文件）、gztar（默认的，gzip 格式的 tar 文件、tar（为压缩的存档文件）、zip（zip 压缩文件）、ztar（compress 命令压缩的存放文件）。

Python setup.py bdist –formats=rpm 可以创建一个 RPM 文件

5.3.3 Windows 版本打包

python setup.py bdist --formats=wininst

这个命令会在 dist 目录下打包成一个 exe 文件，可以直接在 Windows 平台运行，是一个基本的安装向导。

py2exe 是 Distutils 的拓展程序，用来创建可执行的 Windows 程序（.exe 文件），可以在用户不安装 python 解释器的情况下（即默认用户对 python 一无所知）使用。

Python setup.py py2exe 命令在 dist 目录获得 exe 文件。

5.4 自定义源码发布脚本 setup.py

因为 UPlayer 程序涉及的文件和代码量很少，完全可以用一个简单的 python 脚本完成安装设置工作，主要是/usr/share/UPlayer 目录的建立，然后把图标元素图片和默认播放文件分别放到/usr/share/UPlayer/icons 和 media 目录中，最后把 uplayer 和 man 文档分别复制到/usr/bin 和/usr/share/man/man1/中：

```
os.system("cp uplayer.1.gz /usr/share/man/man1/")
os.system("cp UPlayer.py /usr/bin/uplayer")
os.chmod("/usr/bin/uplayer",744)
```

第六章 总结与展望

6.1 本文总结

在 UPlayer 实现的过程中，有过不知所措的困窘，也有攻破关键技术时的喜悦。python 的小巧灵活和 PyQt 强大的功能，为 UPlayer 提供了简洁的 UI 设计。然而，因为对 Qt 的机制并不是特别的了解，而 python 毕竟是解释性语言，这对 UPlayer 的设计难免会有不足之处。不可否认，UPlayer 是一个值得一试的程序

6.1.1 UPlayer 之得

Uplayer 程序的目的就是为 Mplayer 设计一个基于 Linux 的 GUI，这个 GUI 的功能不必和强大，但要提供基本的工作。UPlayer 程序实现对 Mplayer 的播放播放进度控制、声音改变控制、文件更新控制、窗口控制等功能，算得上一个设计简洁、功能丰富的基于 LinuxGUI 视频播放器的实现。

Uplayer 使用 PyQt 的 QProcess 类专心处理 Mplayer 的播放，这样在设计的过程中可以更专注于 UI 的设计，没有偏离预定的设计需求，而去追讨 Mplayer 的播放细节。对于各个快捷键保持了和 Mplayer 的最大兼容性，方便了用户的操作。

对于简洁的 UI 设计，提供的控制元素却很丰富，不但有状态栏，还有工具栏，可以提供更便捷的操作服务。对于初次使用 UPlayer 的用户，或许对 Mplayer 一无所知，但是通过状态栏，可以十分容易的获得各个按钮的使用效果。

简洁的 UI 设计和丰富的功能服务是 UPlayer 的特点。

6.1.2 UPlayer 之失

作为一个视频播放器的 GUI，UPlayer 并没有提供截图、最近播放历史等功能，这可能出于最简单化 UPlayer 的考虑，也处于对这些功能使用频率太低的缘故，取消了这些功能的设计。但是，同一般的 GUI 设计来说，没有这些功能，可以说是 UPlayer 的一个失误，但也正好说明了 UPlayer 的简洁 UI 设计。

对于播放进度条的控制中，通过调试并稳定工作的只有播放进度条控制 Mplayer，而 Mplayer 对时间进度条的控制并不稳定，有时候会出现不一致的情况。而且，对于 QProcess 的 readLine 信息对不同的文件会有些选项不存在或读取不到，问题在于分析过滤程序太过一般化，没有过多考虑特殊情况。对于英文文件名可以很好的给添加到侧边栏，而对于中文，由于不同的编码，会出现乱码情况。有时候在 shell 里面并没有出现乱码，但是到了 GUI 中又出现乱码，这样就只能规定文件名设置为英文名，这样的限制显然是不可取的。

另外，对于全屏/窗口播放、播放/暂停等操作都是通过一个逻辑变量来控制的。这样的设计是不完美的，如果出现一种情况使得这个逻辑变量没有改变，就会出现之后一直以相反状态运行的情况。

6.2 进一步的工作

从 6.1.2UPlayer 之失可以知道 UPlayer 的不足，对 readLine 信息的提取是下一步的重要工作，而对提取出的中文名进行正确的解码将是工作中的重中之重。在 PyQt 中有类 QBtyeArraw、QString、string、char 等类型，通过 QProcess 中提取的文件名是 QString 类型，要转换为 string 类型，现今转换为 QString 类型的方法是 self.movieFile = QtCore.QString(QtCore.QString.fromUtf8(sys.argv[1]))；转换为 string 类型的方法是 unicode(self.movieFile.toUtf8(\),'utf8','ignore').encode('utf8')。在这里添加对字符串编码嗅探的程序，以对中文正确的解码达到在 GUI 中正确显示功能。

另外，Mplayer 的命令控制返回信息很不稳定，需要对 Mplayer 的 slave 模式进行深入研究，弄明白其工作机制。只有这样，才可以达到播放进度条的一致性，同过对时间长度和实际位置进行精确定位。

在时间允许的情况下，可以实现最近播放历史功能，最大存储五条记录，另外要提供清除播放历史功能。

致谢

本文是在姚勇老师和同学的指导和帮助下完成的。最初我对选题要求的理解不够，在姚勇老师的耐心指导下，明白可以使用 MPlayer 来实现解码，并把画面嵌入 GUI 中，对我的设计思路形成有很大帮助，并且在撰写论文时，姚勇老师对论文初稿提出了很多修改意见。另外，很多同学在我撰写此文时对排版等问题上提出了很多宝贵意见。

值此论文结束之际，我以诚挚的心情向他们表示衷心的感谢，感谢他们这段时间里对我的亲切关怀、热情鼓励和悉心指导。

最后，特别感谢我的父母和哥哥给予极大的支持和理解！

参考文献

[1] Mplayer 官网 http://www.mplayerhq.hu

[2] http://blog.csdn.net/liangkaiming/archive/2010/08/05/5791032.aspx

[3] 美，Alex Martelli 著，程胜 杨萍译《Python 技术手册（第二版）》人民邮电出版社 2010 年 6 月

[4]鸟哥编著 许伟 林彩娥改编 《鸟哥的 Linux 私房菜（第二版）》 人民邮电出版社 2007 年 11 月

[5]哲思社区著 《可爱的 Python》 电子工业出版社 2009 年 9 月

[6]法，Tarek Ziade 著 姚军 夏海轮 王秀丽译 《Python 高级编程》 人民邮电出版社 2010 年 1 月

[7]挪，Magnus Lie Hetland 著 司维等译《Python 基础教程（第二版）》 人民邮电出版社 2010 年 7 月

[8]Eric Raymond 《UNIX 编程艺术》

[9]陈儒著 《Python 源码剖析-深度探索动态语言核心技术》 电子工业出版社 2008 年 6 月

[10] 美，Debra Cameron，Bill Rosenblatt，Eric Raymond 著 杨涛 杨晓云 王建桥等译 《学习 GNU Emacs》 机械工业出版社 2002 年 2 月

[11]美，贝尔实验室 Bjarne Stroustrup 著 裘宗燕译《C++程序设计语言》机械工业出版社 2002 年 7 月

[12]Jasmin Blanchette; Mark Summerfield 著《C++ GUI Programming with Qt 4, Second Edition》

[13]Boudewijn Rempt 《GUI Programming with Python: QT Edition》

[14]Brian W.Kernighan ,Dennis M.Richie 著 徐保文 李志译《C 程序设计语言》 机械工业出版社 2004 年 1 月

[15]吴岳等编著 《Linux C 程序设计大全》 清华大学出版社 2009 年 2 月

[16]www.python.org

[17]http://zh.wikipedia.org/wiki/PyQt

[18]http://www.riverbankcomputing.co.uk/software/pyqt/intro

[19]http://www.ibm.com/developerworks/cn/linux/guitoolkit/qt/signal-slot/index.html

附录

在 Debian/Ubuntu 下安装脚本 install.py 如下（完整）：

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
import os
print "=" * 10,"Welcome To UPlayer","=" * 10

print "=" * 5, "mkdir /usr/share/uplayer..."
if not os.path.exists("/usr/share/uplayer"):
    os.system("sudo mkdir /usr/share/uplayer")
print "=" * 5,"done..."

print "=" * 5,"mkdir /usr/share/uplayer/media...."
if not os.path.exists("/usr/share/uplayer/media"):
    os.system("sudo mkdir /usr/share/uplayer/media/")
print "=" * 5,"done..."

print "=" * 5,"mkdir /usr/share/uplayer/icons..."
if not os.path.exists("/usr/share/uplayer/icons"):
    os.system("sudo mkdir /usr/share/uplayer/icons/")
print "=" * 5,"done...."

print "=" * 5,"cp media/* /usr/share/uplayer/media/...."
os.system("sudo cp media/* /usr/share/uplayer/media/")
print "=" * 5,"done..."

print "=" * 5,"cp icons/* /usr/share/uplayer/icons/..."
os.system("sudo cp icons/* /usr/share/uplayer/icons/")
print "=" * 5,"done..."

print "=" * 5,"cp uplayer.man /usr/share/man/man8/...."
os.system("sudo cp uplayer.8.gz /usr/share/man/man8/")
```

```
print "=" * 5,"done..."
print "=" * 5,"you can use 'man uplayer' or 'man 8 uplayer' get the help
manual..."

print "=" * 5,"cp uplayer.py /usr/bin/uplayer....."
os.system("sudo cp uplayer.py /usr/bin/uplayer")
print "=" * 5,"done...."

print "=" * 5,"chmod u+x /usr/bin/uplayer..."
os.system("sudo chmod u+x /usr/bin/uplayer")
print "=" * 5,"done...."
print "=" * 5,"you can use 'uplayer [optin] <movie>' ...."

print "=" * 5,"install python2.6..."
in_python = raw_input("=====Have you installed python?[Y/N]")
if in_python in ["n","no","N","No","NO","nO"]:
    os.system("sudo apt-get install python2.6")
print "=" * 5,"done.."

print "=" * 5,"install pyqt..."
in_pyqt = raw_input("=====Have you installed python-qt4?[Y/N]")
if in_pyqt in ["n","no","N","No","NO","nO"]:
    os.system("sudo apt-get install pyqt-tools pyqt4-dev-tools")
print "=" * 5,"done.."

print '=' * 40
print '=' * 10,"SETUP SUCCESS",'=' * 10
print '=' * 40
```