分类号_____ 密级_____

UDC_____ 编号_____

華中師範大學

硕士学位论文

基于FFmpeg的视频转码与保护系统的设计与实现

学位申请人姓名：张国庆

申请学位学生类别：全日制硕士

申请学位学科专业：教育技术学

指导教师姓名：刘清堂教授 朱晓亮副教授

硕士学位论文

基于 FFmpeg 的视频转码与保护系统的设计与实现

论文作者：张国庆

指导教师：刘清堂 教授　朱晓亮　副教授

学科专业：教育技术学

研究方向：教育信息资源设计与开发

华中师范大学信息技术系

2011 年 5 月

The design and implementation of video transcoding and protection system based on FFmpeg

A Thesis

Submitted in Partial Fulfillment of the Requirement

For the M.S. Degree in Educational Technology

By

Zhang Guoqing

Postgraduate Program

Department of Information Technology

Central China Normal University

Supervisor: Liu Qingtang

Academic Title: professor

Signature Qingtang Liu

Approved

May. 2011

华中师范大学学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的研究成果。除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

作者签名：张国庆　　日期：2011 年 5 月 29 日

学位论文版权使用授权书

学位论文作者完全了解华中师范大学有关保留、使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属华中师范大学。学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密，在 ____ 年解密后适用本授权书。

非保密论文注释：本学位论文不属于保密范围，适用本授权书。

作者签名：张国庆　　导师签名：刘清堂

日期：2011 年 5 月 29 日　　日期：2011 年 5 月 29 日

本人已经认真阅读“CALIS 高校学位论文全文数据库发布章程”，同意将本人的学位论文提交“CALIS 高校学位论文全文数据库”中全文发布，并可按“章程”中的规定享受相关权益。同意论文提交后滞后：□半年；□一年；□二年发布。

作者签名：张国庆　　导师签名：刘清堂

日期：2011 年 5 月 29 日　　日期：2011 年 5 月 29 日

摘 要

互联网应用于教育领域使得信息的传播更加便捷，不同地区的学习者得以共享优秀的教育资源，这极大促进了教育事业的发展。同时，我们也需要面对新的问题。首先，视频播放的终端日益多样化，不同的终端（如手机、PC 机、平板电脑）对视频格式和视频质量的要求也不尽相同。其次，一些未经授权的视频资源流传于网络。一方面，这些盗版视频侵害了视频发布者的利益，打击了优秀教育视频资源发布者的积极性。另一方面，这些盗版视频大部分清晰度不高，内容不连贯，不利于学习者系统高效地进行学习。因此，很有必要设计一种视频转码与保护系统，完成不同视频格式间的相互转换，同时保护正版教育视频不受非法侵害。

源于上述考虑，本文设计了一种基于开源软件 FFmpeg 的视频转码与保护系统，并详细描述了其具体实现过程。论文的主要研究工作如下：

（1） 针对 Linux 环境 FFmpeg 代码的基本解析

通过对 FFmpeg 视频编解码框架的解析，掌握二次开发的基本原则与方法，构建在 Linux 环境下的 FFmpeg 平台，完成支持市面上流行的大部分视频格式之间的相互转换功能。

（2） 转码过程中的视频加密与解密实现

研究如何向视频文件中注入对应的保护信息，以达到未授权的播放器无法播放加密后的视频的效果。研究 ffplay 播放器的播放模块与函数调用流程，在对 ffplay 源码二次开发的基础上实现基于 FFmpeg 的视频数保护系统加密视频文件的回放。

（3） FFmpeg 图形用户界面（GUI)的设计与实现

FFmpeg 基于命令行，不利于操作，本论文将为转码加密模块设计一个基于 Qt 的友好、可视化的图形界面，方便使用者使用。

相比国内外其他研究，本文的特色之处表现在以下两个方面：

（1）将 FFmpeg 应用于数字版权保护领域的研究

本文将 FFmpeg 创造性应用于数字版权保护领域，设计并实现了一个简单有效的视频转码与保护系统。

（2）对 FFmpeg 平台图形用户界面的研究

FFmpeg 是在 Linux 系统下运行的，操作界面是命令行输入。本文为 FFmpeg 设计了一款简洁方便的图形化用户界面，从而使 FFmpeg 的使用变得更加人性化。

关键词：FFmpeg；视频转码；视频保护；加密；解密；Qt

Abstract

The application of Internet in education field makes the dissemination of information to be more convenient. The learners from different regions can share excellent educational resources. This greatly promotes the development of education. At the same time, we also need to face the new problems. First, Some unauthorized video resources spread in the network. On one hand, These video infringe upon the interests of the video publishers. On the other hand, the clarity of these video is not high,the content of these video is inconsistent.These make the learenrs more difficult to learn.Second, the playing terminals are increasingly diversified. Different terminals (such as mobile phone, PC, tablet PC) require different video quality.So, It is necessary to design a video transcoding and protection system.This paper introduces a design of video transcoding and protection system based FFmpeg, describes the detailed its realization process. The main task of this paper is as follows:

(1) The analysis of ffmpeg in Linux development environment

Based on the analysis of ffmpeg frame,we can master the methods of secondary development. This paper constructes FFmpeg platform in Linux environment, completes the conversion between the most popular video formats.

(2) The realization of video encryption and decryption in transcoding process

This paper researches how to inject the corresponding video file digital copyright protection information, researches ffplay's play modules and function invocation process and realizes the decryption of encrypting video.

(3) The GUI's realization of ffmpeg

FFmpeg is based on command line,so it is not convenient in using.This paper designes a friendly GUI based on Qt,make it more convenient to use.

Compared with other research,this paper has two characteristics:

(1) Use ffmpeg in the research of digital copyright protection area

This paper creativly uses ffmpeg in the area of digital copyright protection, devises a simple and effective video transcoding and protection system.

(2)The research on GUI of ffmpeg

This paper designs a concise and convenient graphical user interface for ffmpeg, which makes the use of FFmpeg more humane.

Keywords: FFmpeg; Video transcoding; Video protection Encrypt; Decrypt; Qt

目 录

1 绪论

1.1 研究背景及意义

互联网作为人类历史上最有创造性的发明之一，已经影响到社会生活的方方面面，在教育领域也不例外。目前，互联网的普及使优秀教育信息资源的传播变得简单快捷，人们足不出户就可以享受到优质教师带来的最精彩的教学视频。互联网在给人们带来方便的同时，也面临盗版横行的状况，使得教育视频所有者利益受到不法侵害，这严重损害了资源发布者的积极性。因此，有必要采用内容数字版权加密保护技术对教育视频进行加密，以促进教育信息资源市场的和谐、有序、健康地发展。

目前应用最广的 DRM 系统包括微软开发 Windows Media DRM，网刃，金盾等技术提供商的系统，但均是收费的，且价格昂贵，操作流程复杂，不利于普通消费者购买使用。因此，设计开发一种低成本易操作的视频保护系统显得尤为必要。

视频转码与加密就是将各种格式的原始数据通过视频转换软件的转化，形成统一格式的视频文件。在转码过程中，对视频关键帧进行加密处理，从而完成对整个视频文件的加密。视频解密和播放与上述环节相辅相成。加密后的文件无法被普通播放器播放，只有经过特殊的播放器解密处理后，才可以正常使用。FFmpeg 是一款开源、免费、跨平台的软件，可以用来录制、转换以及流化音视频[1]。FFmpeg 中包含了转码和播放两大基本功能，而且是开源的免费软件，因此，可以在 FFmpeg 基础上进行相应设计，开发出一套视频版权保护系统。

教育视频版权保护是教育信息化得以顺利有序进行的有力保障，本文主要研究视频保护系统中转码和加密解密模块设计与开发，具有重要的研究与应用价值。

1.2 国内外研究现状及存在的问题

1.2.1 国内研究现状

目前国内关于 FFmpeg 方面的论文大多集中在 FFmpeg 的转码应用和嵌入式系统应用方面。

在视频转码方面，已有的研究成果倾向于将 FFmpeg 应用于基于网络的分布式视频转化和发布系统，并有针对性地对 FFmpeg 进行了二次开发以满足特定的需求。如马洪堂的《基于 FFmpeg 的视频转换系统的模块研究》采用 CSCW 计算机支持下

的协同工作理论建立的视频转换系统，并对各个层次所采用的技术进行了深入研究，给出了一个基于 FFmpeg 技术的业务基础架构平台的具体实现[2]。任严等人的《基于 FFmpeg 的视频转换与发布系统》讨论了如何在 Linux 操作系统下，构建一个集视频上传、转换、发布为一身的视频转换系统[3]。单海涛等的《基于 ffmpeg 的高清数字电影软件编码系统的设计》则针对 FFmpeg 不支持非标准 AVI 文件和不支持多个连续文件输出的不足对 FFmpeg 进行了改进，使其转化后的文件可以即可以满足高清数字电影的质量要求，又可以保持原有系统快速高效的优点[4]。

在 FFmpeg 的嵌入式应用方面，国内论文大多是对嵌入式平台搭建、FFmpeg 的移植配置过程和应用方面的研究。如吴张顺《基于 FFmpeg 的视频编码存储研究与实现》介绍了在 video for Linux 接口函数和 FFmpeg 的支持下可以实现实时视频采集与转换嵌入式系统的实现方法[5]。

数字版权保护（DRM）作为一个重要的背景知识，是目前国内外期刊的研究热点。目前，研究主要有数字加密和数字水印两方面的内容。如田虎分析 DRM 的两大关键技术--加密技术和数字水印技术的优缺点，针对这两种技术的缺陷，设计了一种用于保护数字电视业务的基于条件接收和数字水印的 DRM 系统[6]。程春玲则提出了基于模块化实现 DRM 系统的全面解决方案，对 DRM 模型进行了改进，将水印的嵌入移交给权威机构来处理,买卖双方只需对水印的存在性进行检验，实现了对作品的整个生命期的安全访问控制[7]。

1.2.2 国外研究现状

国外对 FFmpeg 研究则较为深入，论文主要集中在 FFmpeg 中一个类库——libavcodec 在其他程序中的应用、FFmpeg 应用于嵌入式系统等方面。代表性的有 Heikki Orsila 等人较好地综述了基于 libavcodec 的开发情况，并对 libavcode 应用于播放器开发和流媒体处理方面进行了相应介绍[8]。Abhishek Udupa 与 Sreepathi Pai 则介绍了对 FFmpeg 进行交叉编译、开展了将 FFmpeg 植入嵌入式处理芯片 Cell BE 的研究[9]。与国内相比，国外版权保护意识更强烈，相应的数字版权保护系统也发展的比较迅速[10]。针对针对不同的电子文档类型，众多公司开发了不同的版权保护系统。通常，电子文档格式主要有两大类型：非固定版式的文件格式和固定版式的文件格式。其中，非固定版式的文件格式包括 html 格式和办公文档保护。例如微软的 office 系列产品中的 DRM 保护服务方案中，作者可以对文档进行加盖水印。在 html 保护反面，微软的 IE 浏览器可以对加密过的 html 进行保护，防止未经授权的

使用者对信息进行复制等操作。固定版式的文件格式常见有 PDF、WDL 等。美国的 Authentica 公司推出了 Secure Documents for PDF 系统，该系统使用 Plug-in 技术对 PDF 进行控制，在国外版权保护市场上享有较高声誉[1]。关于图片的数字版权保护，国外较多的做法是加入数字水印，如 Adobe 公司的著名产品 photoshop 就提供了可以加盖数字水印的相关插件供用户使用。在视音频数字版权保护方面，比较著名的当属微软公司的 Windows Media DRM。该技术用于保护微软的音频（Windows Media Audio 格式，*.wma）和 视频（Windows Media Video 格式，*.wmv）。用户播放受保护的媒体资源时，需要获取一份许可证来解密，方可进行顺利播放。

总体来说，国外关于 FFmpeg 和 DRM 的研究比国内更加深入，这和国外版权保护意识比国内浓厚有很大关系。但相应产品比较复杂，价格昂贵，不适合中国国情。

1.2.3 存在的问题

当前，数字媒体技术已经影响到了人们生活的方方面面。国内外众多的学者纷纷针对开源多媒体领域的鼻祖和王者－FFmpeg 进行了细致的研究。这些研究极大促进了 FFmpeg 技术的发展，但将其应用于视频版权保护领域需要面临新的问题，这些问题主要体现在以下两个方面：

(1)如何寻找将 FFmpeg 应用于视频版权保护领域的研究的切入点

国内外对 FFmpeg 进行研究，绝大多数集中在 FFmpeg 的转码、播放媒体和嵌入式应用等方面。FFmpeg 与视频版权保护的研究相对独立，没有交叉点。针对这个问题，本论文侧重研究 FFmpeg 在视频版权保护领域中的应用，这属于一个新的研究领域，也是本文的特色和创新之处。

(2) 如何实现针对 FFmpeg 平台的图形用户界面的优化

由于 FFmpeg 是在 Linux 系统环境下运行的，所以它的操作是采用的输入命令行的形式。对于习惯了使用 windows 系统的大众来说，这种操作方式相对复杂。这在一定程度上阻碍了 FFmpeg 的发展。国内外的研究对 FFmpeg 平台图形用户界面涉及很少。本文设计了一套基于 Qt 的 FFmpeg 可视化图形用户界面，可以对 FFmpeg 平台图形用户界面的研究起到抛砖引玉的作用。

1.3 论文研究内容

本论文设计了一种简单的视频版权保护系统，实现了基于 FFmpeg 转码的视频

加密与解密，并开发了一种可视化图形界面。本论文主要研究内容如下：

(1)针对 Linux 环境 FFmpeg 代码的基本解析

通过对 FFmpeg 视频编解码框架的解析，把握二次开发的基本原则与方法，构建在 Linux 环境下的 FFmpeg 平台，完成支持市面上流行的大部分视频格式之间的相互转换功能。

(2)基于 FFmpeg 的视频版权保护系统框架研究

提出基于 FFmpeg 的视频版权保护的转码架构，完成系统的概要设计，描述系统的基本功能架构。

(3)转码过程中的视频加密实现

研究如何向视频文件中注入对应的版权保护信息，以达到未授权的播放器无法播放加密后的视频的效果。

(4)播放过程中的视频解密实现

研究 ffplay 播放器的播放模块与函数调用流程，在对 ffplay 源码二次开发的基础上实现基于 FFmpeg 的视频数字版权保护系统加密视频文件的回放。

(5) FFmpeg 图形用户界面的设计与实现

FFmpeg 基于命令行，不利于操作，本论文将为转码加密模块设计一个基于 Qt 的友好、可视化的图形界面，方便使用者使用。

1.4 论文的组织结构

本文组织结构如下：

第一章为绪论，主要介绍研究背景及其意义，以及国内外相关研究现状和存在的问题，并介绍论文的主要研究内容与框架。

第二章主要介绍与研究问题相关的理论与技术基础及 FFmpeg 在 Linux 平台上的配置安装过程，并对 FFmpeg 进行了基本解析。

第三章完成基于 FFmpeg 的视频转码与保护系统框架的设计。在对该系统的受体——视频文件进行分析的基础上总结了系统的基本需求，接下来整个系统按照不同模块分别进行了设计。

第四章完成基于 FFmpeg 的视频转码与保护系统的关键技术实现。主要是转码过程中的视频加密实现与播放过程中的视频解密实现以及转码加密模块的 GUI 实现的具体过程，并对完成后的系统进行了测试。

第五章是全文的总结和下一步工作的展望。

2 相关理论和技术基础

2.1 Linux 系统

Linux 是一类计算机操作系统的统称，是自由软件和开放源代码发展中最著名的例子。Linux 得名于芬兰计算机业余爱好者 Linus Torvalds。它诞生于 1991 年 10 月 5 日。后借助于 Internet 网络，并经过全世界各地计算机爱好者的共同努力，现已成为今天世界上使用最多的一种 UNIX 类操作系统，并且使用人数还在迅猛增长。

Linux 有如下特点：

(1)免费且开源

Linux 是一款免费且开源的操作系统。免费是指使用 Linux 无需向任何人支付任何费用。无论是普通用户版本还是服务器版本或是嵌入式版本，用户均可以从网络免费获得。开源是指 Linux 底层源代码向所有人开放。这样有助于使用者根据自己的喜好对 Linux 对其进行改进。如现在市场上流行的 redhat，ubuntu 和 debian，都是 Linux 爱好者设计对原版 Linux 进行的改进。Linux 的开源特性更有有助于编程爱好者开发基于操作系统底层的软件，这大大促进了自由软件事业的发展[12]。

(2) 适用于嵌入式系统

由于 Linux 系统代码简洁并可随意修改，它常常被应用于嵌入式系统，例如手机、工业控制设备、家用电器等。在智能手机上，Linux 已经成为与 Symbian OS、Windows Mobile 系统并列的三大智能手机操作系统之一。目前市场上流行的 Google 公司 Android 系统就是基于 Linux 内核开发的；很多 CPU 包括家电业芯片，都开始做 Linux 的平台移植工作，其移植的速度远远超过 Java 开发环境。

(3) 安全性

Linux 采取了许多安全技术措施，包括对读、写进行权限控制、审计跟踪、核心授权等技术，这些都为安全提供了保障。由于 Linux 采用的是和 Windows 系统完全不同的设计理念，现有的绝大多数计算机病毒无法对 Linux 系统造成损害。

(4) 易操作性

Linux 有字符界面和图形界面两种工作方式。在字符界面方式下，用户通过输入命令行进行操作。Linux 的图形界面方式已经发展得和 windows 一样方便快捷的地步。如雨林木风（YLMF OS）操作系统就是一个类 windows 十分类似，可以说是一个 Linux 版的 windows 系统[13]。

2.2 视频转码技术

视频转码技术(Video Transcoding)，就是在通过某种手段改变现有视频数据的编码方式，使转码后的视频可以适应不同的网络带宽、不同的终端处理能力和不同的用户需求的一种技术手段[14]。

根据转码的目的不同，视频转码可以分为两类：视频格式转换和视频相关参数改变。视频格式转换顾名思义就是改变视频的封装格式。如为了可以在手机，MP4等设备上观看视频，人们常常将 AVI、RMVB 等格式的视频转换为 3GP 或 MP4 等格式。视频相关参数改变是指改变视频的分辨率，比特率，显示比例等，以压缩视频的大小，达到特定的需要。例如，为节省硬盘空间，人们常常将高分辨率的视频转换成低分辨率的视频。

视频转码技术解码与编码所遵循的视频编码标准可能相同，也可能不同。

不同编码格式之间的视频转码，指通过转码方法改变视频数据的编码格式。通常这种数据转码会改变视频数据的现有码流和分辨率。例如在广播电视领域，大量的数字视频节目采用 MPEG-2 标准编码，但当今许多新的视频播放设备均采用的是 MPEG-4 的解码标准，这样就需要通过转码来讲 MPEG-2 视频解码然后重新编码成符合 MPEG-4 标准的视频。这种转码方式实际就是一个重新编码的过程，因此较为复杂。在实际运用中，经常根据利用所需相互转换编码之间的共同特征来提高转码效率。如在上述例子中，两种编码方式均属于 MPEG 系列标准，均才用了 DCT 变换、MC 运动补偿、MV 运动补偿等关键技术。在转码时，可以利用这些相同点，使转码变得相对简单。

相同编码格式之间的视频转码就相对简单得多。这种视频转码不需要对视频重新进行解码和编码，只是通过转码手段改变其码流或头文件信息就可以完成相互转换。这种转码多用于改变视频相关参数，如将分辨率为 800*600 的视频转换成分辨率为 400*30 的视频，改变视频的比特率等。

2.3 视频保护技术

视频保护技术就是采用特定的加密解密算法来实现对视频的管理与保护，通过加密和一些附带的信息，可以判断出使用者是否有权限来使用、复制和修改对应的视频。视频保护技术一方面可以维护版权市场，使数字媒体作者的权利得到充分保障。另一方面有利于维护通过合法渠道获得数字媒体的用户的合法权益，保证他们享受到高质量的服务，还可以防止经过篡改的视频流入市场，避免对使用者造成不必要的误导[15]。

视频保护主要采用的技术为数据加密和数字水印[16]。

（1）数据加密技术

基于数据加密的视频保护系统工作原理如下:首先对原始文件进行转码，在转码过程中利用密钥对视频进行加密，在原始文件中加入密钥 ID 和授权中心 URL 等信息。完成转码后，通过各种分布方式将视频分发给用户。

用户用视频提供者提供的特定播放器进行播放时，首先通过授权中心的 URL 与授权中心建立联系，然后通过密钥 ID 获取相对应的密钥进行解密，视频方可以正常播放。非授权用户用其他播放器播放时，会出现由于文件格式改变而无法播放的情况，并弹出提示，建议用户联系视频提供者获得视频播放权利[17]。视频保护系统流程如下图 2.1 所示：

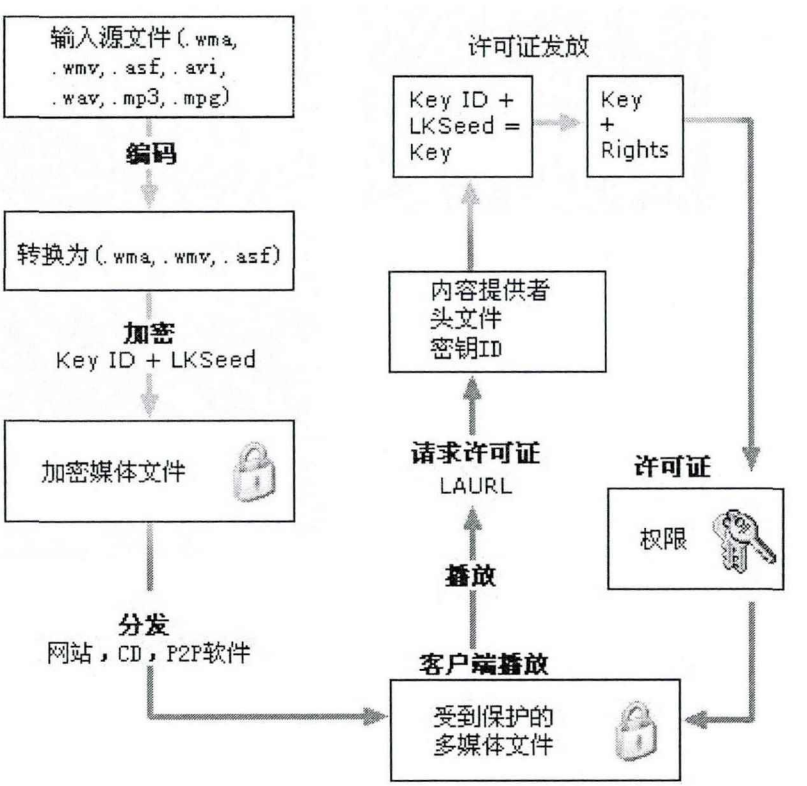

图 2.1 视频保护系统流程

（2）数字水印技术：

数字水印技术是视频保护系统中另一个很重要的技术。数字水印可以分为可见和不可见两种。可见的数字水印一般用于标识视频作品的作者、所有权及来源等，一般为透明或半透明的不影响观看的图形图像。如电视台的台标等。不可见的数字水印肉眼不可见，一般用按某种方式植入到被保护的媒体中。在产生版权纠纷时，

版权所有者就可以利用相应算法提取到该数字水印，来证明自己对产品的著作权[18]。

作为数字水印技术具有下面几个特点：

- 隐藏性

数字水印以不影响被保护数据正常使用为原则，而且数字水印不应该影响被保护数据的品质。

- 鲁棒性

鲁棒性是指数字水印应该具有很强的健壮性，具有可以抵御有意识或者无意识的破坏的能力。如果想去除水印，必须对被保护数据进行相当大的修改，甚至这种修会破坏数据的完整，影响到数据的正常使用。

- 明确性

数字水印应该可以明确证明被保护数据的版权归属。当版权发生纠纷时，数字水印算法应该能准确提取出被嵌入数据中的保护信息，将其作为解决版权纠纷的重要依据。

数字水印是视频保护的最后一道屏障。它不能保护数据被非法使用传播，但可以为鉴定被保护数据的所有权，追踪非法侵权的源头。数字水印技术往往和数据加密技术一起使用，两者互取所长，共同构建视频保护的屏障[19]。

2.4 图形用户界面

GUI 是图形用户界面的缩写(Graphical User Interface)。与 GUI 相对应的是命令行界面。在早期的 DOS 操作系统和现在的 Linux 系统中，均采用了命令行程序。命令行程序具有占用系统资源少，效率高和安全性好等优点，但复杂繁琐的各项输入参数却让不少用户生畏[20]。 命令行界面如下图 2.2 所示：

图 2.2 命令行界面

GUI 用户界面弥补了命令行界面的不足。由于 GUI 界面美观生动，具有很强的

人机交互性，用户无需专门训练既可以掌握软件的使用。它是一种融合了计算机科学、教育心理学、行为科学、美学以及各领域需求分析的系统工程，其目的就是减轻使用者负担，使操作更加便捷，从而提高产品的竞争力。GUI 界面如图 2.3 所示：

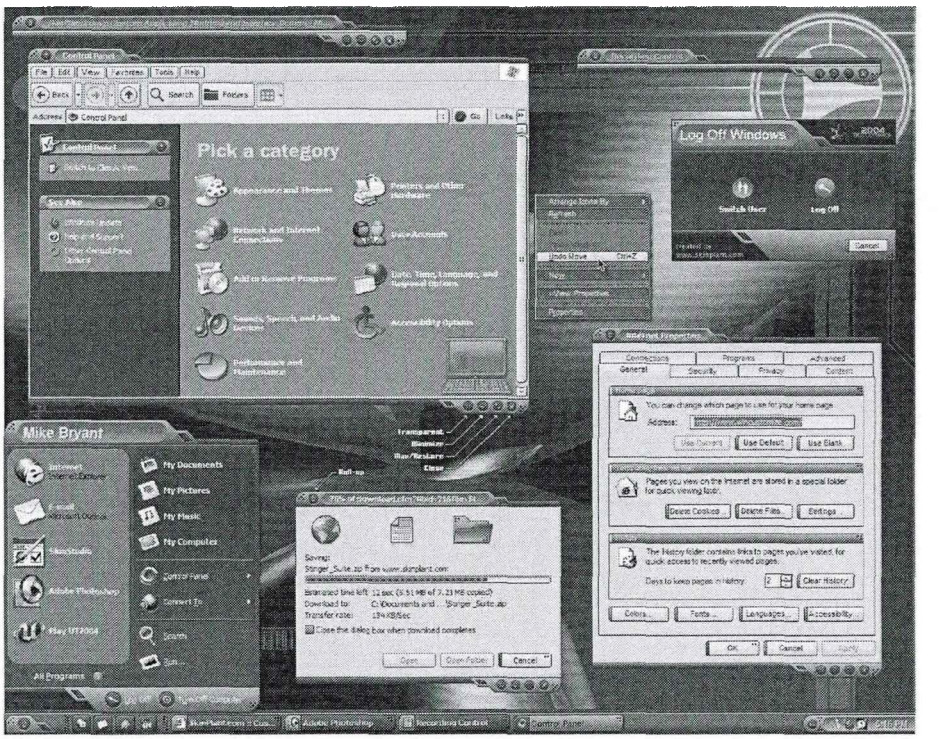

图 2.3 GUI 界面

2.5QT

2.5.1 Qt 概述

Qt 是一个由挪威的 TrollTech 公司开发的跨平台的 C++图形用户界面应用程序框架。2008 年 1 月，通讯巨头诺基亚公司成功收购了 TrollTech 公司，引进了其旗下包括 Qt 在内的所有技术，并根据移动平台规划需要，对 Qt 进行了不断的完善[21-22]。Qt 发行版本有自由版、嵌入式自由版、企业版和专业版。其中，自由版和嵌入式自由版是免费的，分别面向 Unix/X11 系统和嵌入式系统。企业版和专业版是诺基亚面向商业用户提供的版本，是收费软件。

2.5.2 Qt 特性

作为一种优秀的跨平台的 C++图形用户界面应用程序框架，Qt 具有以下特点

[23]：

（1）跨平台性：

Qt 采用“一次编写，到处运行”的方法支持开发跨平台的 GUI 应用程序，这是 Qt 一个突出优势。使用单一源码树和简单的重编译方式，Qt 可以在如下平台运行：

- MS/Windows - 95、98、NT 4.0、ME、2000、xp 和 7
- Unix/X11 - Linux、Sun Solaris、HP-UX、Compaq Tru64 UNIX、IBM AIX、SGI IRIX 和其它很多 X11 平台
- Macintosh - Mac OS X
- Embedded - 有帧缓冲(framebuffer)支持的 Linux 平台。

（2）面向对象性

Qt 采用了面向对象编程思想，所有函数依据功能被划分为若干模块，这些模块之间都存在着相应的继承关系。因此，Qt 函数具有很高的重复利用性。这大大减轻了程序开发者的负担。

便于进行图形开发

Qt 提供了专门用于图形开发的工具——QtDesigner。QtDesigner 类似于网页制作工具 Dreamweaver，用户可以用拖动控件方式设计图形界面，使得开发更加直观快捷。

（3）支持 2D 与 3D 图形开发

Qt 通过 QPainter 和 QGraphicsView API 对高级 2D 图形提供了支持。QPainter 提供了一个全面的 2D 绘图框架。除了渲染多边形、绘图路径、访射和非访射变形体这些基本功能之外，它还支持平滑处理、渐变画刷和 alpha 混合。Qt Graphics View 提供了 2D 场景图形 API。它带有高级 API，用于在场景中放置对象（形体）并在多个视图中显示该场景[24]。

此外，Qt 通过其 OpenGL 模块支持 3D 图形开发。Qt 开发人员可使用 OpenGL 在 GUI 应用程序中绘制 3D 图形。Qt 提供了单独的 Qt OpenGL 模块，可将 OpenGL 图像与本地窗口系统集成在一起。

（4）支持各类数据库

针对 Oracle、Microsoft SQL Server、Sybase Adaptive Server、IBM DB2、PostgreSQL.、MySQL、Borland Interbase、SQLite 和 ODBC 兼容的数据库，Qt 提供了本地驱动。另外，Qt 还提供了专用于数据库的控件，使任何内建或自定义控件均可感知数据。

2.5.3 Qt 模块

Qt 由模块组成，Qt 中的类库根据功能不同划分为若干大类模块。当设计和部署 Qt 程序时，设计者可以根据自己的需要选择特定的模块，这样可以使部署 Qt 程序更加快捷。Qt 同时提供了一些只针对特定操作系统的模块，如 QtDBUS 是针对 unix 系统，而 QtAxContainer 和 QtAxServer 只针对 windows 系统[24]。Qt 模块如下图 2.4 所示：

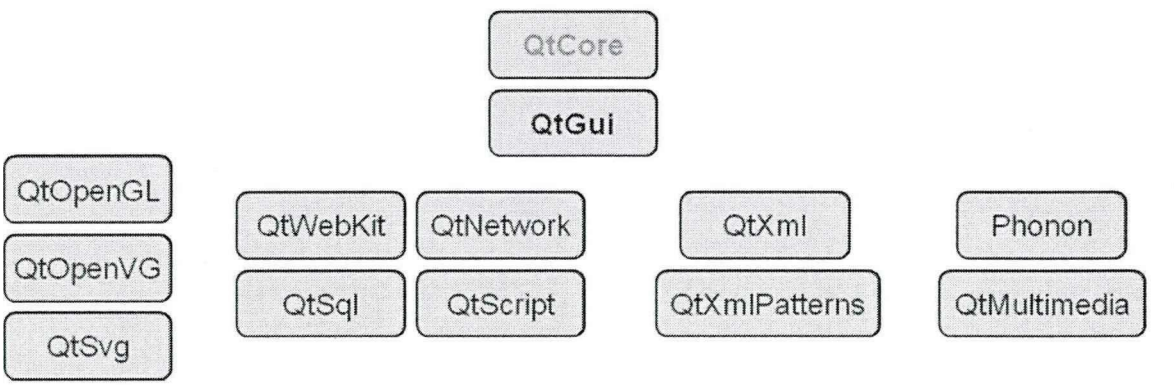

图 2.4 Qt 模块图

2.5.4 Qt SDK 的安装

Qt SDK 是诺基亚公司开发的全新 Qt 软件开发包，包含了 Qt 库、Qt Creator IDE 和其他 Qt 工具，Qt 安装过程如下表 2.1 所示：

表 2.1 Qt SDK 安装

1	登陆 Qt 官方网站 http://qt.nokia.com/downloads/sdk-Linux-x11-32bit-cpp 下载最新版本 SDK。最新版本为 qt-sdk-Linux-x86-opensource-2010.05.bin。
2	打开一个 SHELL，即终端，输入“su”然后输入 root 密码，成为 root 用户
3	用 CD 命令进入 bin 文件所在的目录
4	在终端输入: chmod u+x qt-sdk-Linux-x86-opensource-2010.02.bin。 说明：chmod 命令用来提升用户对文件的操作权限，以便对该文件进行操作。
5	在终端输入下列命令：./qt-sdk-Linux-x86-opensource-2010.02.bin ，则出现安装界面。

6	按照提示进行安装，安装过程和 windows 下执行 exe 安装程序类似，可以修改安装路径等。本设计将其安装在/usr 文件下。
7	在/etc 目录下找到 profile 文件，终端输入“vi”进入文件编辑模式，然后输入 I 键进入编辑模式（也可用软件 gedit 对其进行编辑），在 profile 最后位置加入下面四行： QTDIR=/usr/qtsdk-2010.05/qt PATH=$QTDIR/bin:$PATH LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH export QTDIR PATH LD_LIBRARY_PATH 其中第一行是 qt 软件安装后所在的路径，第二行是 qmake 可执行程序的路径，第三行是 Qt 中库的路径，第四行是 Qt 的输出路径。
8	在终端执行:qmake -v 命令就可以看到 qmake 的版本信息，运行 qtcreator 就可以看到开发界面了，至此 Qt SDK 安装成功[25] 。

2.6 FFmpeg

2.6.1 FFmpeg 简介

FFmpeg 项目由法国天才程序员 Fabrice Bellard 发起，其目的是开发一个开源的多媒体播放系统（类似于 Mplayer）。FFmpeg 项目提供了一个集视音频采集、格式转换、音/视频编码解码功能为一体的完整的开源解决方案。FFmpeg 基于 Linux 操作系统开发，但是可以在大多数操作系统（windows、Mac OS）中编译和使用[26]。目前其最新的版本是 0.6.1 版本。用户可以通过 svn 获得最新版的源码，地址为 svn checkout svn://svn.ffmpeg.org/ffmpeg/trunk ffmpeg。用户也可以通过 http 下载方式通过搜索引擎下载最新版本的源码。FFmpeg 支持 AC3、DV、FLV、MPEG、DivX、MPEG4 等 40 多种编码，AVI、MPEG、OGG、MP3、ASF 等 90 多种解码。它被 mplayer 使用作为解码器，国内比较流行的播放器影音风暴的后端 ffdshow 也是使用 FFmpeg 的解码库。英文“FF”代表“Fast Forward”，“mpeg”表示其符合 MPEG 视频编码标准。FFmpeg 中包含了转码和播放两大基本功能，而且是开源的免费软件，因此，可以在 FFmpeg 基础上进行相应二次开发，研制出一套视频版权保护系统。

FFmpeg 基本框架如表 2.2 所示：

表 2.2 FFmpeg 的基本框架

基本模块	功能描述	其它属性
ffmpeg	视音频转换器、	命令行工具 ，用来对视讯档案转换格式，也支援对电视卡即时编码
ffserver	支持视频和音频的流媒体服务器	支持 RTP/RTSP/HTTP 协议
ffplay	简洁的媒体播放器	用来测试 FFmpeg 中的各种 API 函数的功能
libavcodec	FFmpeg 音频和视频 encode 和 decode 函数库	支持常见的音视频编解码
libavformat	音频和视频 muxer 和 demuxer 函数库	Muxer 模块实现将视频文件、音频文件和字幕文件的合并为某一个视频格式，如可以将 aa.avi、bb.mp3 和 cc.srt 组成一个 dd.mkv 文件。Demuxer 的作用与之相反，是将视频容器中的视频流，音频流和字幕等分离出来
libavuti	小型函数库	包括 CRC 校验码的产生;最大公约数，整数开方；整数取对数；内存分配 [27]

上表中，ffmpeg 不仅支持不同格式之间的相互转化，还可以改变包括视频的比特率、帧频、帧大小、音频采样率等等在内的数十个参数。ffserver 默认在无交互后台模式下运行。当在调试模式或在有非后台参数输入的情况下，它也可以转到前台运行。libavcodec 与 libavformat 一起构成整个项目的核心。ffplay 利用了 FFmpeg 里面的类库和 SDL 中的类库。界面比较简单，支持全屏、快进、快退，但没有进度条控制播放进度。用户可以在此基础上进行改写，从而完成一个功能完善的播放器，是开发其他播放器的一个很好的参考[28]。

2.6.2 FFmpeg 搭建

（1） 搭建 Linux 系统环境

由于 FFmpeg 是基于 Linux 环境构建的，所以搭建 FFmpeg 环境的第一步就是

构建 Linux 系统。构建 Linux 环境可以有三种方式：

- 在 PC 机上安装 Linux 系统开发环境

这种方式是在 PC 裸机或安装了 windows 操作系统的 PC 机上安装 Linux 系统。Linux 不同版本的源码可以从网上直接免费下载。当前世界上有超过三百个版本存在，其中以 Ubuntu、Redhat、Debian 和 Suse 最为出名。国内也有红旗，雨林木风等适合中国用户的版本。不同版本的 Linux 各有特色，用户可以根据自己的喜好选择相应的版本安装。

这种方式的 Linux 环境稳定且安全性好，并可以安装众多软件使开发工作更加便捷，是进行 Ffmpeg 开发的首选。

- 在 windows 系统中安装虚拟机

虚拟机（Virtual Machine）是一个想象中的计算机，是指通过软件方式模拟的具有现实中计算机所有功能的虚拟计算机。通过它，用户可以在一台实体 PC 机上虚拟出若干个虚拟 PC 机。这些虚拟 PC 机具备真实计算机的所有功能，用户可以在虚拟机上安装 Linux 系统，安装相应的 Ffmpeg 软件。目前流行的虚拟机软件主要有 Vmware 和 Oracle xVM VirtualBox[29]。

通过这种方式构建的 Linux 系统具备和真实系统一样的功能，可以方便 windows 用户随时切换两种操作系统。不足之处是虚拟机运行速度受实体 PC 机性能影响较大，并且虚拟的系统不是特别稳定。

- 通过 Msys+Mingw 构建类 Linux 环境

Msys 是指 Minimal GNU（POSIX）system on windows，是 windows 环境下模仿 Linux 开发环境的最好的一个软件。利用它可以在 windows 环境下用 Linux 命令行进行操作，可以代替 cmd。

Mingw 即 Minimalist GNU For Windows。它是一些头文件和端口库的集合，该集合允许人们在没有第三方动态链接库的情况下使用 GCC（GNU Compiler C）产生 Windows32 程序。

Msys+Mingw 是安装在 windows 系统下的一个类 Linux 开发环境，即相当于在 windows 环境下构建一个可以使用 Linux 命令行的编译系统和命令行终端。这个环境下，用户可以无需安装 Linux 系统直接用 windows 使用 make 命令编译在开源软件。同时具有良好的通用性和兼容性。这两个软件都是免费软件，与之相对应的也有很多实用工具，并支持版本更新。不足之处是 Msys+Mingw 只是一个编译环境，不能安装其他的一些在 Linux 环境下运行的辅助性开发工具，开发起来会有些不便。

安装 Msys+Mingw 步骤如下：

下载最新版 Msys 安装程序，将其安装在 D 盘。

下载 Mingw 安装程序，注意要把其安装在 Msys 文件夹下的 Mingw 文件夹中。（Msys 安装好后会在其目录下生成一个 Mingw 文件夹）

修改 msys.bat，用记事本程序打开 Msys 目录下的 msys.bat 文件，将用户机上 vcvars32.bat 实际存在的路径写入到 msys.bat 文件。vcvars32.bat 位于 vc 目录下，将其写入 bat 文件是为了使用微软的工具产生 lib 文件。如某台机器 vcvars32.bat 位于“D: \Microsoft Visual Studio 8\VC\bin”，则应该在 bat 文件后写入

call "D:\Microsoft Visual Studio 8\VC\bin\vcvars32.bat"

系统整合

在 msys 目录下找到 fstab.sample，将其改名为 fstab。用记事本程序打开，把这行：C:/mingw /mingw 改为 d:/msys/1.0/mingw/mingw32。

下载 pr.exe 并安装（如不安装的话系统会报错）。pr.exe 是一个权限提升管理工具[30]。

以上三个 Linux 各有利弊，使用者可以根据自己偏好选择相应的开发环境。

（2） FFmpeg 的配置与安装

安装好 Linux 环境后，下一步就是配置和安装 FFmpeg。

- Linux 环境下安装软件类型及安装方式

Linux 环境下软件安装与 windows 中的软件安装有很大的不同。Windows 环境直接双击安装文件即可以安装，Linux 中的安装文件则有以下几种格式：后缀为.rpm 最初是 Red Hat Linux 提供的一种包封装格式，现在许多 Linux 发行版本都使用；后缀为.tar.gz、tar.Z、tar.bz2 或.tgz 是使用 Unix 系统打包工具 tar 打包的；后缀为.bin 的一般是一些商业软件[31]。

RPM 安装软件：

RPM 全称是 Red Hat Package Manager（Red Hat 包管理器），这种软件包和微软操作系统中的 EXE 文件类似，通过鼠标左键双击的方式就可以实现软件的安装。

使用源代码安装软件：

使用源代码安装软件是 Linux 环境下安装软件一个很重要的手段，也是 Linux 的一个很主要的优势。使用这种方式安装软件，有利于按照用户的需求有针对性的安装软件，而不仅仅是按照安装包中预定的参数进行安装。安装流程为：

1、打开一个 SHELL，即终端，输入“su”然后输入 root 密码，成为 root 用户

2、用 CD 命令进入源代码压缩包所在的目录

3、根据压缩包类型解压缩文件(*代表压缩包名称)

tar -zxvf ****.tar.gz

tar -jxvf ****.tar.bz(或 bz2)

4、用 CD 命令进入解压缩后的目录

5、输入编译文件命令：./configure

6、然后是命令：make

7、再是安装文件命令：make install

8、安装完毕

BIN 文件安装：

扩展名为.bin 文件是二进制的，它也是源程序经编译后得到的机器语言。BIN 文件安装和 RPM 文件很相似，在用“chmod +x”命令取得了对该 BIN 文件进行操作的权限后，就可以像 windows 下安装 EXE 文件一样对该文件进行安装了。安装流程为：

1、打开一个 SHELL，即终端，输入“su”然后输入 root 密码，成为 root 用户

2、用 CD 命令进入源代码压缩包所在的目录

3、给文件加上可执行属性：chmod +x ******.bin（中间是字母 x，小写）

3、执行命令：./******.bin(qt for Linux 就是这样的安装包)

- FFmpeg 安装过程

FFmpeg 安装程序是后缀名为.tar.gz 或 tar.bz2。软件包里面是原程序，没有经过编译，必须要经过编译才可以安装。

在安装 FFmpeg 之前，首先要安装外部的支持库。在不安装外部支持库的前提下，FFmpeg 只支持实现 flv 向 avi、asf、 mpeg 的转换或者将 avi、asf、 mpeg 转换为 flv。如果要想让 FFmpeg 支持更多的格式，就必须安装相应的外部支持库[32]。外部支持库及 FFmpeg 配置安装过程如下：

- 外部库安装

支持 mp3

要想使 FFmpeg 支持 MP3 格式，必须首先安装 lame 的获取地址如下：

http://lame.sourceforge.net/index.php

下载完成后，打开终端，输入超级用户口令获得 root 权限，用 cd 命令切换到存放下载文件的目录下，然后用 tar-zxvf 命令解压.tar.gz 文件，然后执行./configure、make 和 make install 命令。例如下载的 lame 安装源代码为 lame-3.97.tar.gz，存放路径为/usr/bin,则在终端中输入的命令为：

1、tar -zxvf lame-3.97.tar.gz

2、cd /usr/bin/lame-3.97

3、./configure --enable-shared --prefix=/usr

4、make

5、make install

其中--prefix 后面指定的路径为 lame 的安装路径，可以更改。

支持 MP4 格式

MP4 格式是目前手机，MP4 等便携视频播放设备支持的最常用格式，FFmpeg 默认不支持这种格式的文件，必须下载相应的外部库。

获取地址：FAAD2 http://www.audiocoding.com/modules/mydownloads/

FAAC　http://prdownloads.sourceforge.net/faac

安装编译过程和 lame 过程相似，获取 root 用户权限后，在终端中输入代码如下：

FAAD2

1、autoreconf -vif

2、./configure --prefix=/usr --with-mp4v2 --enable-shared

3、make

4、make install

FAAC

1、tar zxvf faac-1.26.tar.gz

2、cd faac

3、./bootstrap

4、./configure --prefix=/usr

5、make

6、make install

支持 3GP 格式

下载 amr-wb 包,解压，编译，安装，在获取 root 用户权限后，在终端输入命令行如下：

1、wget http://ftp.penguin.cz/pub/users/utx/amr/amrwb-7.0.0.1.tar.bz2

2、tar jxvf amrwb-7.0.0.1.tar.bz2

3、cd amrwb-7.0.0.1

4、./configure --prefix=/usr/local

5、make

6、sudo make install

下载 amr-nb 包,解压，编译，安装，在获取 root 用户权限后，在终端输入命令

行如下：

1、wget http://ftp.penguin.cz/pub/users/utx/amr/amrnb-6.1.0.4.tar.bz2

2、tar jxvf amrnb-6.1.0.4.tar.bz2

3、cd amrnb-6.1.0.4

4、./configure --prefix=/usr/local

5、make

6、sudo make install

以上只是列出了三种得到外部库的支持后 FFmpeg 可以支持的格式，如果想让 FFmpeg 支持更多格式，可以查阅 FFmpeg 项目组的文档。

➢ FFmpeg 编译和安装

在外部库安装完成以后，就可以进行 FFmpeg 的安装了。FFmpeg 的安装过程和上面安装外部库的过程大体相同。

1、首先下载 FFmpeg 源码。用户可以通过 svn 获得最新版的源码，地址为 svn checkout svn://svn.ffmpeg.org/ffmpeg/trunk ffmpeg。用户也可以通过 http 下载方式通过搜索引擎下载最新版本的源码。

2、打开一个 SHELL，即终端，输入“su”然后输入 root 密码，成为 root 用户。

3、用 CD 命令进入源代码压缩包所在的目录

4、解压缩文件

tar -zxvf ffmpeg0.6.1.tar.gz

5、/configure --prefix=/usr --enable-shared --enable-mp3lame --enable-amr_nb --enable-amr_wb --enable-faad --enable-faac

6、make

7、make install

在步骤 5 的编译文件过程中，有很多编译选项，上面提到的参数是针对前面提到的外部库来进行设置的。在 ffmpeg 源文件夹下，输入./confirure －h 选项，可以看到 FFmpeg 提供的可供选择的数百行参数，这就是开源软件的优势所在。用户可以“订制”安装软件，而不拘泥于系统已经设置好了的安装选项（如 windows）。

上述步骤完成后，进入/usr 路径下，可以看到 ffmpeg 图标存在。打开终端后输入“ffmpeg －v”命令，可以看到当前 ffmpeg 的版本，至此，ffmpeg 安装过程结束。FFmpeg 安装完成后的版本信息如图 2.5 所示：

```
$ ffmpeg -v
FFmpeg version 0.6, Copyright (c) 2000-2010 the FFmpeg developers
  built on Oct 15 2010 18:08:19 with gcc 4.5.0
  configuration: --enable-shared --disable-static --enable-memalign-hack
  libavutil     50.15. 1 / 50.15. 1
  libavcodec    52.72. 2 / 52.72. 2
  libavformat   52.64. 2 / 52.64. 2
  libavdevice   52. 2. 0 / 52. 2. 0
  libswscale     0.11. 0 /  0.11. 0
C:\msys\1.0\local\bin\ffmpeg.exe: missing argument for option 'v'

Administrator@CCHUZGQ ~
```

图 2.5　FFmpeg 版本信息

2.6.3 FFmpeg 参数解析

FFmpeg 功能十分强大，除了视音频格式转换外，它还具有屏幕图像录制、视频抓图、视频采集等等许多十分实用的功能。在终端输入“ffmpeg　-h”命令，可以查阅其帮助文档[FFmpeg 工程组]。

（1）FFmpeg 选项

FFmpeg 的帮助文档提供了数百个输入选项，其强大的视音频处理功能由此可见一斑。常用输入选项可以分为如下几类：

➢ 通用选项

通用选项包括提取软件许可证、显示帮助信息、显示版本信息、设置文件作者标题等等。部分通用选项如下：

-L license

-h 帮助

-fromats 显示可用的格式，编解码的，协议的

-f fmt 强迫采用格式 fmt

-I filename 输入文件

-y 覆盖输出文件

-title string 设置标题

-author string 设置作者

-copyright string 设置版权

-comment string 设置评论

➢ 视频选项

通过输入相应的视频选项，使用者可以在转码过程中改变视频的一些参数，如改变比特率、设置帧速率、设置视频横纵比等等。部分视频选项如下：

-b bitrate 设置比特率，缺省 200kb/s

-r fps 设置帧频 缺省 25

-s size 设置帧大小 格式为 WXH 缺省 160X128.

-aspect aspect 设置横纵比 4:3 16:9 或 1.3333 1.7777

-croptop size 设置顶部切除带大小 像素单位

-padtop size 设置顶部补齐的大小 像素单位

-maxrate bitrate 设置最大视频码率容忍度

-minrate bitreate 设置最小视频码率容忍度

➢ 3 音频选项

通过输入相应的音乐频选项，使用者可以在转码过程中改变视频的一些参数，如改变比特率、设置帧速率、设置视频横纵比等等。部分视频选项如下：

-ab bitrate 设置音频码率

-ar freq 设置音频采样率

-ac channels 设置通道，缺省为 1

-an 不使能音频纪录

-acodec codec 使用 codec 编解码

（2） FFmpeg 使用语法

FFmpeg 基本使用语法规则如下：

ffmpeg -i INPUTfile [OPTIONS] OUTPUTfile

其中，INPUTfile 是指输入文件的名称，OUTPUTfile 是经过转码后输出文件的名称。通过读取这两个选项，FFmpeg 可以调用与其格式相匹配的解码器和编码器来完成编解码工作。[OPTIONS]就是上面提到的可选择的输入选项，可以改变输出文件的视音频参数。例如，如果想把一个名为 ccnu 的 avi 文件转化为同名的 flv 文件，并把该 flv 的比特率设为 100kb/s。则在应在终端中输入如下命令：

ffmpeg –i ccnu.avi –b 100 ccnu.flv

2.7 本章小结

本章主要介绍了和研究问题密切相关的理论与技术基础，分别介绍了 Linux 系统的特点、视频转码与保护主要采取的技术、GUI 图形界面的定义和优点、Qt 用户界面应用框架与 FFmpeg 的基本框架和 Linux 环境下 FFmpeg 的编译与安装。

3 基于 ffmpeg 的数字版权保护系统设计

本章首先从全局的角度对视频文件相关背景知识进行说明，然后详细阐述了系统需求分析与总体设计，随后分模块对系统进行了详细的设计。

3.1 视频文件相关背景知识概述

3.1.1 容器格式和编码格式

人们设定了不同的视频文件格式来把视频和音频放在一个文件中，以方便同时回放，实际上就是把不同种类的多媒体数据流（多为视频流和音频流）合并在一个单一的文件内，这个文件就叫容器。容器是用来区分不同文件的数据类型的，而编码格式则由音视频的压缩算法决定，通常所说的文件格式或者是后缀名指的就是文件的容器，如 AVI、RMVB、MP4、3GP 等等。

视频编码方式就是指通过特定的压缩技术，将某个视频格式的文件转换成另一种视频格式文件的方式[33]。常用的视频编码方式有国际标准化组织运动图像专家组的 MPEG 系列标准运动，静止图像专家组的 M-JPEG 和国际电联的 H.263/H.264 标准，此外在互联网上被广泛应用的还有微软公司的 WMV、Real-Networks 的 RealVideo 以及 Apple 公司的 QuickTime 等[34]。

例如：将一个 Xvid 视频编码文件和一个 MP3 视频编码文件按 AVI 封装标准封装以后，就得到一个 AVI 后缀的视频文件，这个就是我们常见的 AVI 视频文件了。

由于很多种视频编码文件、音频编码文件都符合 AVI 封装要求，则意味着即使是 AVI 后缀，也可能里面的具体编码格式不同。因此会出现在一些设备上，同是 AVI 后缀文件，一些能正常播放，还有一些就无法播放的现象。

常见容器与编码格式组合如下表 3.1 所示：

表 3.1 常见容器与编码格式组合

常见容器与编码格式组合		
封装容器	视频流编码格式	音频流编码格式
AVI	Xvid	MP3
AVI	Divx	MP3
Matroska（MKV）	Xvid	MP3
Matroska（MKV）	Xvid	AAC
Matroska（MKV）	H264	AAC

MP4	Xvid	MP3
MP4	H264	AAC
3GP	H263	AAC

3.1.2 流，包和帧

一个视频中一般包括一个视频流，一个音频流和一个字幕流。“流”是一种形象的说法，用来表示一连串用时间串连在一起的数据元素。

视频流、音频流和字幕流组成了一个视频文件。在 FFmpeg 里面，系统是调用 muxer/demuxer 来拆分和组合这些流的。以 avi 为例，此种格式的 muxer/demuxer 位于 libavformat 文件夹下，分别为 avienc.c 和 avidec.c。

avi 的 demuxer 定义（AVInputFormat avi_demuxer）如下表 3.2 所示:

表 3.2 avi 的 demuxer 结构

字段定义	字段属性
avi	格式名
NULL_IF_CONFIG_SMALL("AVI format")	格式长名称
sizeof(AVIContext)	所对应的私有结构的大小
avi_probe	接口函数，测试传入的数据段是否是符合当前文件格式
avi_read_header,	接口函数，读取头文件
avi_read_packet,	接口函数，读取包
avi_read_close	,接口函数，关闭文件
avi_read_seek	接口函数，寻找文件位置

avi 的 muxer 定义（AVOutputFormat avi_muxer）如下表 3.3 所示：

表 3.3avi 的 muxer 结构

字段定义	字段属性
"avi"	格式名
NULL_IF_CONFIG_SMALL("AVI format")	格式长名称
"video/x-msvideo"	MIMIE Type,即资源类型
"avi"	后缀名
sizeof(AVIContext)	所对应的私有结构的大小

CODEC_ID_MP2	音频编码 ID
CODEC_ID_MPEG4	视频编码 ID
avi_write_heade	接口函数，书写头文件
avi_write_packet	接口函数，书写包
avi_write_trailer	接口函数，书写文件尾

在 libavformat 文件夹下，部分 muxer/demuxer 如下表 3.4 所示：

表 3.4 部分 muxer/demuxer 头文件与源文件

部分 muxer/demuxer	
头文件	源文件
asf	asfenc asfdec asfcrypt
avi	avienc avidec
matroska	matroskaenc matroskadec
flv	flvenc flvdec

每个流是由不同的编码器来编码生成的。编解码器描述了实际的数据是如何被编码 Coded 和解码 DECoded 的，这些解码器与编码器被称为 decoder 和 encoder。Divx 和 MP3 就是编解码器的例子[原理]。

在 FFmpeg 中，视音频的编解码器都位于 libavcodec 文件夹下，当进行视频转码时，系统就会调用相应的编解码器完成视音频流的编码解码工作。

部分 decoder/ encoder 如下表 3.5 所示：

表 3.5 部分 decoder/ encoder 头文件与源文件

部分 decoder/ encoder	
头文件	源文件
libxvid_internal	libxvid_rc libxvidff
mjpeg	mjpeg_parser mjpega_dump_header_bsf
mpeg4video	mpeg4videodec mpeg4videoenc
h264	h264_cabac h264_direct
aac	aaccoder aacenc

流中的基本数据元素称为“帧”Frame。从流中被读出来的叫做“包”Packets。包是一段数据，它包含了一段可以被解码成方便我们最后在应用程序中操作的原始帧的数据。根据我们的目的，每个包包含了完整的帧或者对于音频来说是许多格式的完整帧。在本系统中，核心工作就是对包和帧进行处理，已达到视频加密解密的目的。一个帧被相应保存在一个包或多个包中[36]。

3.1.3 IPB 帧

MPEG 运动图像编码技术标准是由 Motion Picture Experts Group 在 1988 年提出，并于 1992 年 11 月通过，1993 年 8 月作为 ISO/IEC 11172 标准公布，这就是通常所说的 MPEG-1。MPEG-1 为了追求更高的压缩效率，更注重去除图像系列的时间冗余度，因此引入了 I 帧(帧内编码)、P 帧(前向预测编码)、B 帧(双向预测编码)。P 帧由前一个 I 帧或 P 帧图像来预测，而 B 帧由前后的两个 P 帧或一个 I 帧和一个 P 帧来预测，因而编解码和帧的显示顺序有所不同。

仅仅使用前一个显示的基准帧来编码的帧被称为“P 帧”，同时使用前一个显示帧和未来帧作为基准帧进行编码的帧称为“B 帧”。在通常的场景中，编解码器编码一个 I 帧，然后向前跳过几个帧，用编码 I 帧作为基准帧对一个未来 P 帧进行编码，然后跳回到 I 帧之后的下一个帧。编码的 I 帧和 P 帧之间的帧被编码为 B 帧。之后，编码器会再次跳过几个帧，使用第一个 P 帧作为基准帧编码另外一个 P 帧，然后再次跳回，用 B 帧填充显示序列中的空隙。这个过程不断继续，每 12 到 15 个 P 帧和 B 帧内插入一个新的 I 帧[37]。表 3.6 很直观地表现了视频中帧的编解码顺序与显示顺序。

表 3.6 视频中帧顺序

	帧的显示顺序													
帧的编码（解码）顺序	I													
					P									
		B												
			B											
				B										
									P					
						B								
							B							
								B						
													P	
										B				
											B			
												B		
														I

因此可以推断出，视频帧是有关键帧和非关键帧之分的，I 帧即为关键帧，P、B 帧为非关键帧。非关键帧可以由关键帧预测得到。所以，对视频进行加密，只需对关键帧进行加密即可。一旦关键帧加密后，非关键帧也就相当于自动进行了加密。解密时也是基于同样的道理。所以，处理关键帧是对视频进行加密解密的关键所在。

3.1.4 MPEG 系列标准

MPEG 有两种含义，一种含义是指 Moving Picture Experts Group 的简称，即是指 1988 年成立的“运动图像专家小组”这个组织，另外一种含义是指这个组织所指定的一系列视频编码标准。其中，第二种含义更为人所知。在本文中，MPEG 是指第二种含义，即一种是视频编码标准。

到目前为止，该标准一共有六个成员：

MPEG-1：发布于 1992 年，有三层，其中第二层用于 VCD 的视频编码，第三层多用于音频编码，即常见的 MP3。

MPEG-2：发布于 1994 年，通常用来为广播信号提供视频和音频编码，包括卫星电视、有线电视等。这种编码方式可以提供广播级别的视频图像和 CD 级别的音质效果。DVD 就是采用了 MPEG-2 标准来进行编码的。

MPEG-3：MPEG-3 本为高清电视（HDTV）设计，因 MPEG-2 在此领域表现十分突出，所以中止了研发工作。

MPEG-4：发布于 1998 年，这个标准包含了 MPEG-1 与 MPEG-2 的绝大部分功能，并吸收了当时网络上流行的其他视频编码方式的长处。由于采用了特殊的基于对象识别的编码方式，采用 MPEG-4 标准进行编码的图像占用空间小，图像质量佳，并可以做到在低带宽条件下传输视频并保证视频质量。

MPEG-7：MPEG-7 其实并不是一种视频压缩编码方法，它重点在于影音内容的描述和定义，以明确的资料结构和语法来定义影音资料的内容。使用者可以通过这个标准有效搜索自己想要的影音资料。

MPEG-21：MPEG-21 是一个正在制定中的标准，目的就是理解如何将不同的技术和标准结合在一起需要什么新的标准以及完成不同标准的结合工作。

MPEG 系列视频编码标准为了使压缩比更高，将视频分为关键帧与非关键帧，即非关键帧可以通过由关键帧通过数学预测的方式得到。帧间压缩编码技术、DCT 技术和熵编码是该系列标准采用的主要编码技术[38]。

3.2 系统需求分析

当今社会，信息化浪潮席卷了社会生活的方方面面，教育领域也不例外。互联网技术的迅速普及，更进一步促进了教育信息化的发展。互联网技术使教育信息资源的传播变得简单方便。只要有条件连上互联网，用户不论处于乡村还是城市，都可以及时获得教育发达地区最优秀的教育资源[39]。

互联网技术的发展在给教育信息化发展带来新契机的同时，也带来了一些负面的问题。一些未经授权的视频资源流传于网络，这些视频资源良莠不齐。一方面，这些盗版视频侵害了视频发布者的利益，打击了优秀教育视频资源发布者的积极性。另一方面，这些盗版视频大部分清晰度不高，内容不连贯。这也不利于学习者系统高效地进行学习。因此，采用数字版权保护系统，维护教育视频资源的合理有序进行，是很有必要的。

目前的数字版权保护系统价格昂贵，系统实现比较复杂，显然不适合我国教育信息化的具体情况。所以，很有必要设计一套简单廉价的数字版权保护系统，来为教育视频资源的合理有序传播提供保障。

数字视频版权保护系统主要有三部分组成：文件转码加密模块，证书引擎模块和解码播放模块。作为整个系统的第一个组成部分，文件转码加密模块在整个系统中处于基础性和决定性的地位。证书引擎模块为整个系统提供加密技术支持，是维持系统有效性的重要保障。解码播放模块面向用户，是整个系统得以存在推广的最重要的环节。

3.3 系统总体设计

3.3.1 系统功能

视频转码与加密系统应具有的功能有：

（1） 可以实现常见视频格式之间的转化。

系统应具有处理多格式视频相互转换的功能。一方面，视频来源广泛，格式多样化情况普遍存在。另一方面，视频使用者对视频格式有不同的需求，有的用户希望视频质量清晰，有的用户则希望视频体积小（如 3GP，MP4 格式）。这两方面都对系统的视频格式兼容性提出了较高的要求[40]。

（2）加密效果良好。

经过该系统处理后的文件，普通播放器不能播放，需得到数字中心授权后放可以观看使用。加密算法要求抗解密能力强，加密文件要求理论上不能被解密或解密

要求时间过长。

（3）播放解密效率高。

在用具有解密功能的播放器播放加密视频时，应能做到事实解密，流畅播放。除播放加密视频外，播放器还应该具有播放其他非加密视频的功能，以满足用户的不同使用需要。

3.3.2 系统的功能框图

根据以上分析， 一个简单的视频数字版权保护系统应具有以下三个模块：用户界面模块，转码加密模块和播放解密模块。

用户界面模块为整个系统提供了一个友好度高，美观简洁的图形界面，弥补了FFmpeg默认的命令行界面所带来的不足。

转码加密模块可以完成不同视频格式之间的相互转换，并可以在转码的同时完成视频加密工作。输入视频通过这一模块后，就变成了特定格式的加密视频。

播放解密模块在接收加密视频后，经过对加密视频进行相应的处理，可以将加密视频还原成普通视频，并可以对其进行正常播放，同时，该模块还具有播放普通视频的功能。

系统整体框架如下图 3.1 所示：

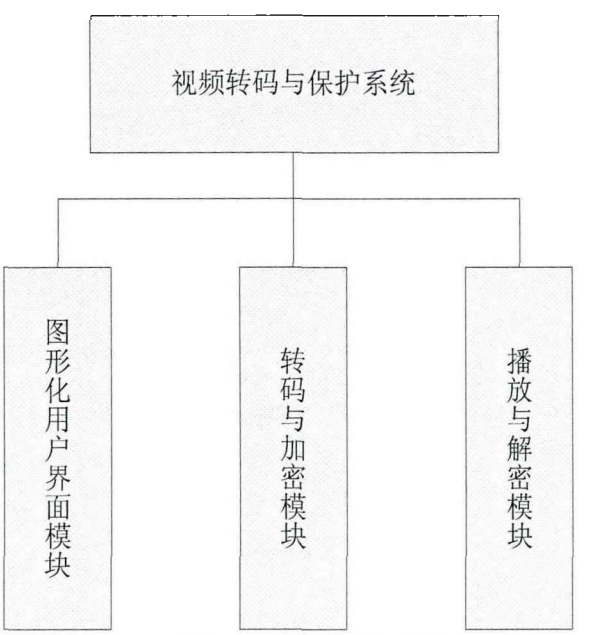

图 3.1 系统框架图

3.4 系统功能模块设计

3.4.1 转码加密模块设计

转码加密模块是该系统核心模块。该模块主要功能有两项：完成不同格式视频间相互转换和视频加密。这两项功能里面，FFmpeg 已经具备了对不同视频格式进行相互转换的功能。所以，视频加密功能又是该模块的核心。

FFmpeg 处理视频的流程是先将输入文件逐帧解码，然后按照输出格式文件的要求重新进行编码。视频加密函数应该加在帧解码后与编码前。这种加密算法不破坏视频文件头，不会都视频造成损害。视频帧进行加密，这样就不容易被轻易解密。

在 MPEG-1、MPEG-2、MPEG-4 和 H.264 这些视频编码算法中，视频帧是有关键帧和非关键帧之分。通过对关键帧进行相应的运算，就可以得到非关键帧，所以，在对视频加密过程中，只需要对关键帧部分进行加密即可。关键帧加密后，非关键帧也就相应进行了加密。这样有助于提高视频加密效率，也有利于解密播放器端流畅运行。

视频文件一般由视频流，音频流和字幕流组成，其中最重要的是视频流。视频流加密以后，即使音频流与字幕流是完好的，整个视频文件也无法被播放。所以，只要完成了对视频流的加密，也就相当于完成了整个文件的加密。所以，本系统中只对视频流进行加密处理，对音频流和字幕流暂不处理。这样的处理方式，工作量相对较小，但同样可以达到预期的效果。

视频帧的解码编码工作涉及到 FFmpeg 底层算法，相对比较复杂，所以，在进行加密处理时，尽量不要涉及和修改 FFmpeg 底层的编解码算法。视频转码由三个步骤组成，解码，编码和将编码后的写入文件，在本系统中，具体加密函数放在编码后和写入文件前进行。

具体来说，在 FFmpeg 中，Avpacket 作为写入文件的基本单元而存在，也就是说，经过视频帧是存放在 Avpacket 中写入到输出文件中去的。Avpacket 是一个结构体，定义如下：

```
typedef struct AVPacket {
    int64_t pts;            //显示时间戳
    int64_t dts;            //解码时间戳
    uint8_t *data;          //帧数据
    int     size;           //实际保存音视频数据缓存的大小
```

```
    int     stream_index;        //音视频数据包对应的流索引
    int     flags;               //关键帧标志
    int     duration;
    void (*destruct)(struct AVPacket *);
    void *priv;
    int64_t pos;
} AVPacket;
```

其中，uint8_t *data 是一个指针，指向保存在内存中的视频帧。对视频帧进行加密，就是对 uint8_t *data 所指的视频帧进行相应的处理。在 FFmpeg 中，write_frame(AVFormatContext*s,AVPacket*pkt,AVCodecContext*avctx,AVBitStreamFilterContext *bsfc)函数具体作用就是将编码后数据包中的帧写入输出视频文件，而它的传入参数之一就是存放着视频帧的 AVPacket。所以，在 write_frame 运行前，对 AVPacket 中的成员变量 data 进行加密，就可以得到对整个视频文件进行加密的效果。此外，还应该为 AVPacket 结构体增添一个成员变量 int encrypt，该成员变量用于记录完成加密后，将其赋值为 1。这样，就可以再后面的解码过程中根据这个变量的值来判断是否应该对某一帧进行解密，以免造成将未加密的正常视频进行解密情况的出现。

针对视频帧，本系统采用异或算法进行对其进行加密，异或是一种二进制的位运算，它具有一个非常有用的性质：自反性。即 A XOR B XOR B = A，对给定的数 A，用同样的运算因子（B）作两次异或运算后仍得到 A 本身。

在加密时，采用异或算法对数据帧与一串字符进行异或运算，得到了加密后的数据帧。在解密时，再次采用异或算法与相同的字符串进行运算，就会得到原来的数据帧[41]。

按位进行异或运算是计算机最为常用的运算方式之一，相对其他加密算法，异或运算方便快捷，计算机执行效率高，而且方便编写解密算法，所以，在本系统中，对视频帧的加密和解码是用异或运算来实现的。

该模块流程设计如下：首先解码器工作，对输入文件逐帧进行解码，解码完成后，系统判断此帧是否为关键帧，如果是关键帧，则对该帧进行加密，如不是，则不进行处理。转码加密模块流程如下图 3.2 所示：

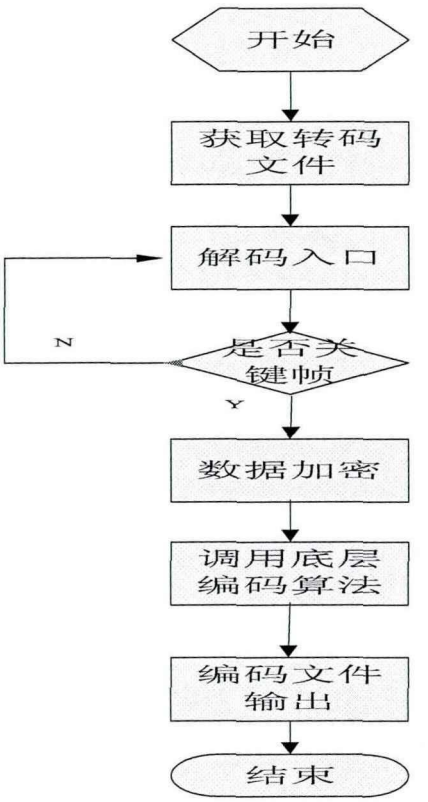

图 3.2 转码加密模块流程图

对关键帧进行加密是该模块的重点。在本模块中，先将解码后的视频帧从内存中提取出来，然后对其进行加密处理，处理完成后，数据重新进行编码，生成加密后新文件。

该模块转码与加密一起进行，转码完成后，加密工作也相应完成。加密后的文件，普通播放器无法播放或播放质量严重受损。必须用特定的播放器，得到相应解码信息后方可以进行播放。

3.4.2 播放解密模块

播放解码模块是视频数字版权保护系统中最后一个模块，该模块直接与用户产生联系，在整个系统中占有很重要的地位。

本模块基于 FFmpeg 中自带的 ffplay 播放器进行开发。ffplay 利用了 FFmpeg 里面的类库和 SDL 中的类库，主要用来测试 FFmpeg 中的各种 API 函数的功能。界面比较简单，支持全屏、快进、快退，但没有进度条控制播放进度。

在前面的视频转码加密模块中，加密视频必须经过相应的解密程序解密后方可

以进行播放。因为前面的视频文件加密是对视频关键帧进行的加密，所以相应的解密工作也应该对视频关键帧进行解密，这样才可以使视频还原，方便进行播放。

另外，播放器还应具有播放正常视频的功能，所以要首先判断当前的视频是否为加密视频，如果是，则进行解密运算然后播放，如果不是，则直接进行播放。

该模块与上面的转码加密模块相辅相成，原理相近。在播放解码模块，同样需要对数据结构 AVPacket，进行处理。首先要从中读出 data，然后对 data 进行异或运算，得到解密后的 data，最后将 data 传入下一个步骤，从而完成视频的解密工作。

播放解码模块工作流程如下：获取到加密文件后，播放器首先判断读取的帧是否为关键帧，如果是关键帧，则对其进行解密，然后进行解码，最后进行播放。播放解密模块工作流程如下图 3.3 所示：

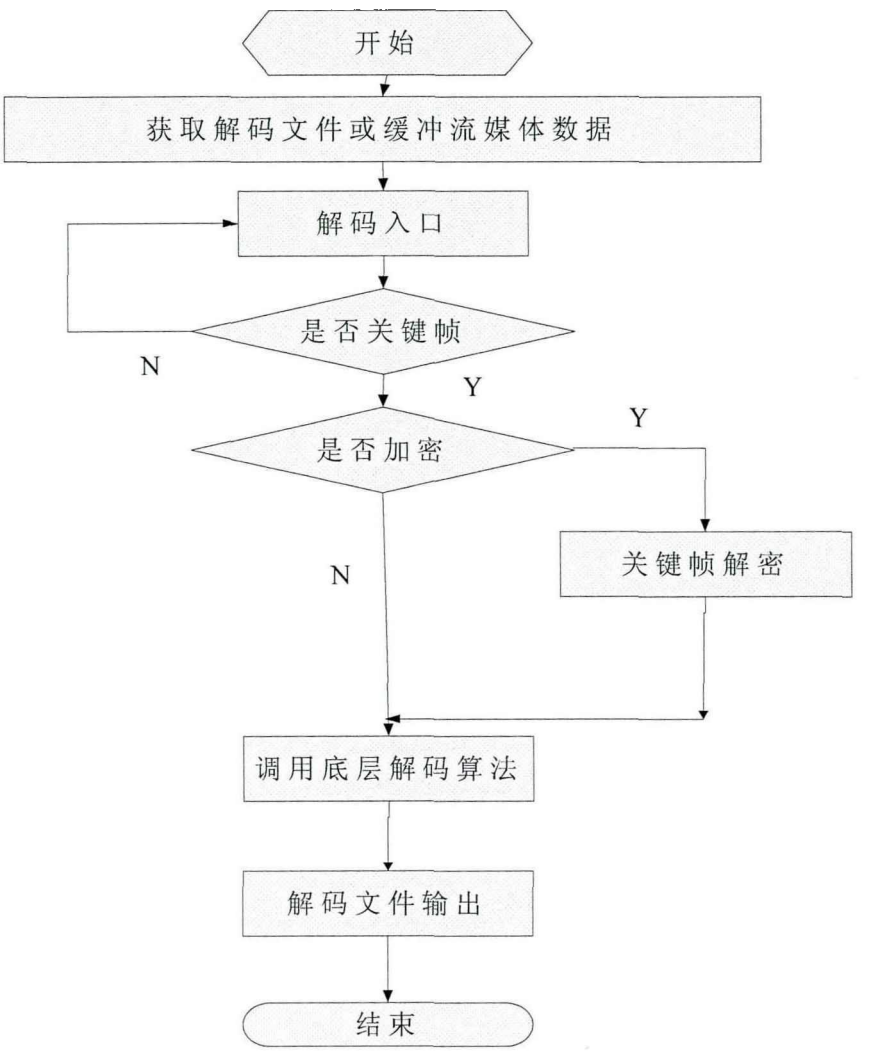

图 3.3 播放解密模块流程图

3.4.3 用户界面模块设计

用户界面层的功能就是为用户和底层 FFmpeg 软件之间提供一个方便传递参数的桥梁。该系统的用户界面应具有如下特点：

（1）界面简洁美观。

UI 界面优于命令行界面之处在于具有良好的用户友好性，赏心悦目，便于使用。因此，该界面必须美观大方，能给用户带来愉悦的使用体验。FFmpeg 中命令行参数十分复杂，不可能都在一个界面上有所展示。本设计选取了里面最常用到的若干个基本参数，通过这些参数，可以完成大部分的视频转码工作。

（2）功能实用

用户通过在该界面上进行操作，可以方便完成大部分不同格式的视频转码工作，还可以视频帧速率，比特率等参数进行设置。针对特定的播放硬件（如手机，MP4），界面应该自动设置好转化后的视频格式（3GP、MP4）和视频参数。

（3）模块化设计

该界面采用模块化设置，界面层源代码应该具有面向对象的特性，代码复用性好。当需要对界面进行改善时，只需要改动很少代码就可以完成改进工作。

（4）和 FFmpeg 源码相对独立

界面层和转码层加密层应相对独立。界面层与 FFmpeg 源码之间，只存在一个函数用于命令行参数传递。界面层代码与 FFmpeg 代码相对独立，界面层的改动不会对转码加密工作带来影响。

综上所述，系统用户界面层总体设计图如下图 3.4 所示：

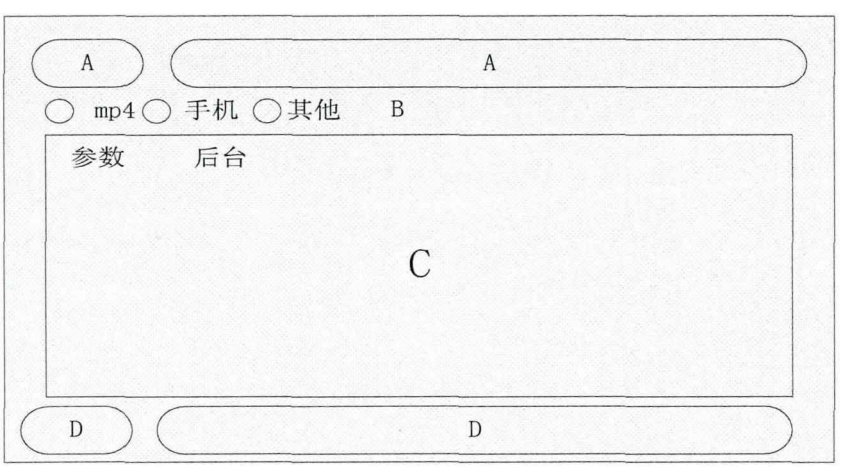

图 3.4　总体设计图

A 处为输入文件选择区。点击按钮，可以根据存储路径选择输入文件。

B 处为特定格式选择区。用户可以选择“MP4”、“手机”或“暂无”选项，对

应的输出文件分别是 MP4、3GP 和自由选择格式。

C 处是参数选择与后台输出区。当用户在 B 处选择“MP4”或“手机”格式时，该处会自动提示出最佳视频参数设置。参数设置也可以手动输入，以满足不同需求。此处还提供另一个选项卡，可以输入 FFmpeg 后台运行情况。

D 处是输出文件区。当用户在 B 处选择选择“MP4”和”手机”格式时，该处会自动设置成对应的 MP4 和 3GP 格式输出文件。当在 B 处选择“暂无“格式时，该处可以手动设置输入文件的格式。

该界面采用 Qt SDK 设计，使用 C++为开发语言，代码重复利用率高。同时与底层相对独立，不影响 FFmpeg 转码加密层运行。

3.5 本章小结

本章首先对该系统的受体——视频文件的相关背景知识进行了介绍，然后对系统进行了需求分析，接下来对整个系统按照转码用户界面模块，转码加密模块和播放解码模块来分别进行了设计。

4 视频转码与保护系统的实现

本章具体阐述如何实现视频转码与保护系统即视频转码加密与视频播放解密功能。

4.1 系统开发环境

本系统在 Linux 操作系统环境下开发，具体版本为 Ubuntu 5.10。

使用 FFmpeg 版本为 FFmpeg- 0.6。本系统可视化用户界面给予 Qt SDK 开发，具体版本为 qt-sdk-Linux-x86-opensource-2010.05。

4.2 转码加密模块实现

4.2.1 视频转码处理流程

FFmpeg 对视频进行转码处理大体流程如下：

（1）注册所有容器格式和编解码器

主要包括 avcodec_register_all()、avdevice_register_all()和 av_register_all()三个函数，avcodec_register_all 注册 coded\decoded 及硬件加速器，avdevice_register_all()注册硬件设备，av_register_all()注册 demuxer/muxer 及 protocol。

（2）匹配合适的 demuxer 和 muxer

demuxer 和 muxer 根据输入文件和输出文件的后缀名来进行匹配。具体来讲 demuxer 的匹配是首先由函数 opt_input_file()解析 void parse_options(int argc, char **argv, const OptionDef *options)中的“-i”。而 muxer 是由函数 guess_format 根据 main( ) 函数的 argv 里的输出文件后缀名来进行的。

上述注册和初始化 demuxer 与 muxer 是在 ffmpeg 的 main 函数中进行的，接下来的过程主要是在函数 av_transcode 中进行，也就是说，av_transcode（旧版本中为 av_encode）是整个 ffmpeg 的核心函数，大部分的转码工作都会在这个函数里面进行。

（3）匹配合适的 decoder 和 encoder

decoder 和 encoder 分别由函数 AVCodec *avcodec_find_encoder(enum CodecID id)和函数 AVCodec *avcodec_find_decoder(enum CodecID id)来完成匹配，这两个函数根据传入的 CodecID id 找到相应的 encoder 和 decoder，然后保存到 AVInputStream 和 AVOutputStream 所属的变量中。AVStream 结构保存与数据流相关的编解码器，数据段等信息，包括两个成员变量 AVCodecContext *codec 与 void *priv_data。

（4）对视频文件的每一帧进行转码操作

这一步是转码工作的核心。FFmpeg 会对输出文件格式做一个判断，如果输出文件格式是 RAW 图像（即 YUV 或 RGB）那么就没有编码函数，直接写入文件。如果是非 RAW 图像，则要求对文件进行逐帧编码，然后写入文件。

（5）关闭各个 demuxer 和 muxer，decoder 和 encoder，释放内存。

这个过程和开始的匹配 demuxer 和 muxer，decoder 和 encoder 相反[42]。

4.2.2 主要函数解析

（1）av_transcode

av_transcode 函数是整个转码过程中的最为重要的函数，它完成了转码的大部分工作。

对视频进行加密处理也应该以这个函数为切入点。av_transcode 位于 ffmpeg.c 文件中，其流程基本步骤包括：

- 输出输出流初始化
- 解码器和编码器初始化
- 从输入文件中获取元数据信息并进行处理
- 检查输出文件的参数是否符合规范，如不符合则报错
- 书写输出文件头部信息
- 对输入文件的每个帧，首先读取该帧，然后解码帧数据，接着编码帧数据，最后将编码后的帧数据写入输出文件
- 书写输出文件尾部信息
- 释放内存，注销掉各个 demuxer 和 muxer，decoder 和 encoder

在上面过程中，步骤 6 是最为核心的一个过程，完成了帧解码，编码和写入文件等一系列的操作，其余步骤都是为它服务的。

（2）write_frame

阅读 av_transcode 源代码，发现 av_transcode 函数是通过调用 output_packet() 函数的方式具体实现了步骤 6 功能。

output_packet()函数原型位于 ffmpeg.c 文件，首先是判断输入视频时候是否需要解码，如果需要，则分别对视频流，音频流和字幕流进行处理。然后判断输出视频时候需要重新编码，如果是，则重新编码，然后将其写入至视频输出文件。

仔细分析 output_packet()代码，在其中寻找关键函数 write_frame，发现共有三

个地方调用。其中有两个是通过函数 do_video_out 调用的，其余两个是在函数本身调用。这样，就完成了 write_frame 函数的定位。FFmpeg 系统调用函数 write_frame 的流程如下图 4.1 所示：

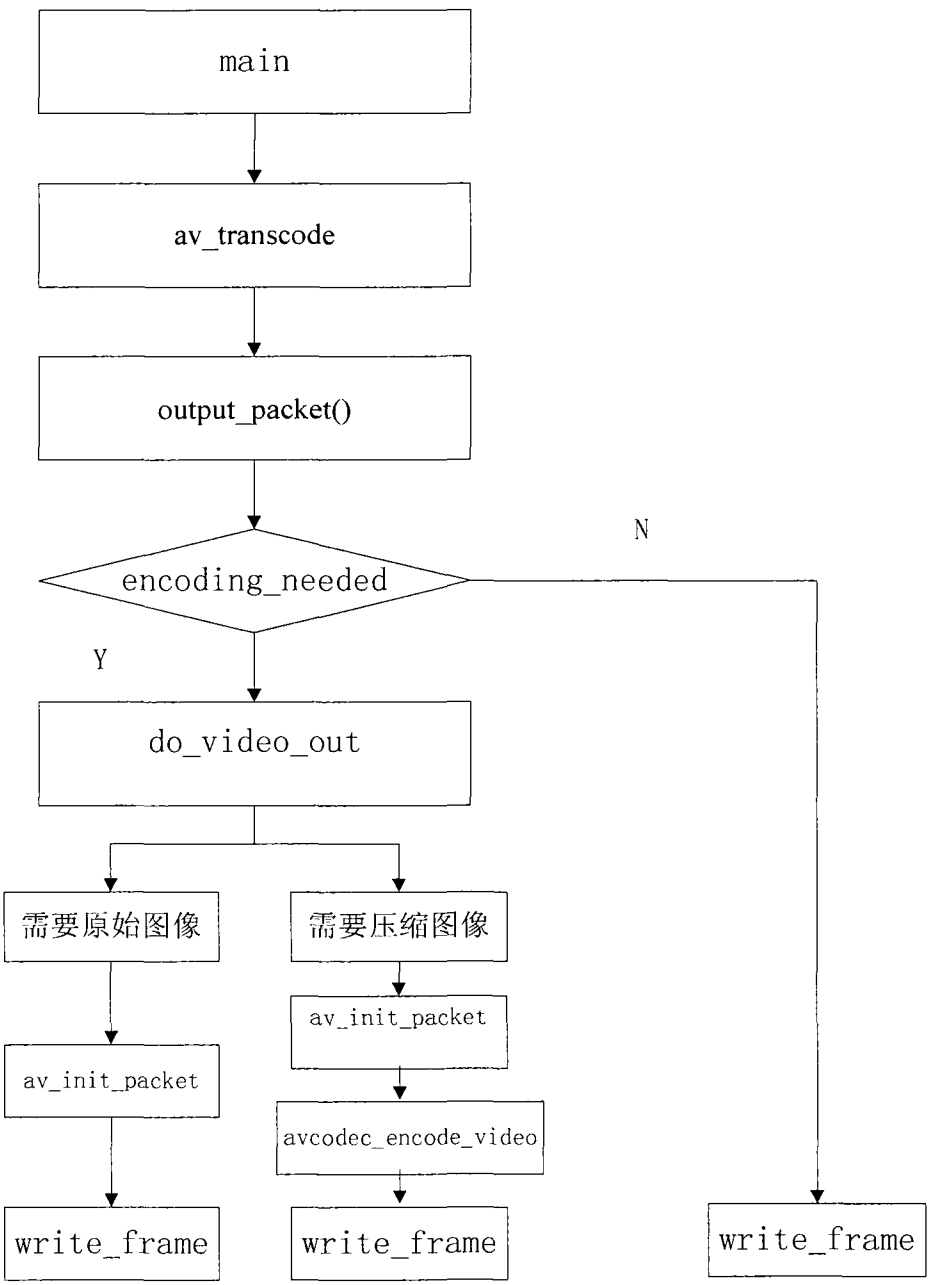

图 4.1　函数 write_frame 流程

4.2.3 具体加密函数实现

本系统采用异或算法对视频帧进行加密，首先将 AVPacket 里面的 data 变量转化为字符串，然后将其与另外一个字符串进行异或运算。

本系统采用的异或算法本质是将 data 变量转换为字符串，然后将其与另外一个字符串（即 key）进行按位异或运算，用伪代码描述如下;

str 表示将要被加密的字符串，key 表示密钥。sPos 与 kPos 表示字符串 str 与密钥 key 的字符成员所在位置，初始值为 0，即都从字符串的第一位开始。

（1）在 sPos 小于字符串长度的情况下

IF kPos 大于等于密钥长度，则置 kPos 为 0

从 0 位开始对 str 与 key 进行按位以后运算

sPos 与 kPos 加一，嵌套调用 xorEncrypt 函数

（2 ）IF sPos 大于字符串长度（按位异或运算已经完成）

返回经过处理后的 str 值

异或算法关键代码如下：

```
static char *xorEncrypt(char *str, char *key, int sPos, int kPos)
{
    if (sPos < strlen(str))
    {
        if (kPos >= strlen(key))
        {
            kPos = 0;
        }
        str[sPos] = str[sPos] ^ key[kPos];
        return xorEncrypt(str, key, ++sPos, ++kPos);
    }
    else
    {
        return str;
    }
}
```

将异或加密函数加入到函数 out_packet 和 do_video_out 所调用的 write_frame 前面：

```
if(pkt.flags==1)//判断是否为关键帧
```

```
{
  pkt.data=(uint8_t *)xorEncrypt((char *)(pkt.data),"asdfghjk",0,0);//异或运算
  pkt.encrypt=1;//已经加密
 }
write_frame(s,&pkt,ost->st->codec,bitstream_filters[ost->file_index][pkt.stream_index]);
```

其中，pkt.flags==1 是用来判断该包中的视频帧是够为关键帧，是的话，就对其进行加密，如果不是，则不进行任何处理。"asdfghjk"为加密字符串，相当于密钥，在加密与解密过程中用有着非常重要的作用。这里只是一个例子，根据需要，可以更换更为复杂的字符串，达到更好的加密效果。将 pkt.encrypt 设置为 1 是为了给加密视频文件做一个标记，这样后面的播放器模块可以根据此标记来判断是否应该对播放的文件进行解密处理，以避免对正常文件进行解密处理情况的发生。

按照上述方式对 ffmpeg.c 文件进行修改后，进行正常编译安装，所得到的 ffmpeg 系统就可以对视频文件进行加密了。

4.3 播放解码模块实现

4.3.1 ffplay 播放器模块

ffplay 播放器基本上由五个模块组成，它们分别是：源文件模块、解复用模块、解码模块、颜色空间转换摸块和渲染模块。其基本模块如下图 4.2 所示：

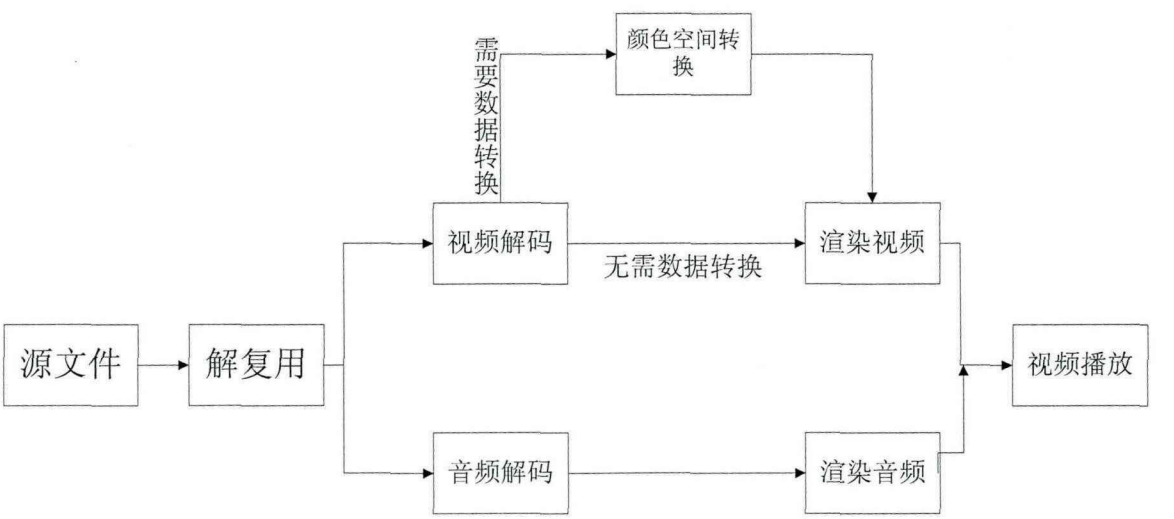

图 4.2 ffplay 播放器的基本播放模块

其中，源文件模块是这个播放器的起始，主要是为下面的各个模块以数据包的

方式提供数据流，对 ffplay 而言，其作用是从本地视频文件中读取出数据包，然后将其按照一定的顺序排列，源源不断得发送到下面的解复用模块中。在 ffplay 中，此模块主要是通过调用 libavformat 文件夹下的 file.c 等文件来运行。

解复用模块根据源文件的容器格式来分离出视频流，音频流和字幕流，在加入时间同步等信息后传送给下面的解码模块。这个过程与 ffmpeg 中解复用过程基本相似。其涉及到的文件主要是 libavformat 文件夹下的的 avidec.c，utils_format.c 文件。

解码模块作用是对数据包进行解码处理，将数据包中保存的帧转换成 YUV 或 RGB 数据，并将上面解复用模块中的时间同步信息传送下去。这个模块涉及到的文件主要是在 libavcodec 目录下。

颜色空间转换模块主要完成将解码出来的数据转换成当前系统支持的颜色数据的功能。颜色数据有 YUV 和 RGB。一般解码器解码出来的都是 YUV 数据。当前的 PC 机所用的显卡都支持这两种颜色数据的，但有些嵌入式系统或比较老旧的显卡只支持 RGB 格式的数据。颜色空间转换模块就是在必要的时候将 YUV 数据转化成 RGB 数据。此模块需要的文件包括 libavcodec 目录下的 imgconvert.c 等文件。

渲染模块对视频来说就是显示视频图像，对音频来说就是播放声音，对字幕来说就是显示字幕，并保持视频、音频和字幕的同步播放。在 ffplay 中，这一模块通过调用 SDL 库来实现[43]。

对于视频数字版权保护系统的播放器模块而言，主要的开发工作应该是对放在解码模块中来进行的。对于前面经过加密的关键帧，在播放端，应该在解码模块中对其进行解密处理，将其还原成正常的帧。对于 ffplay 的其他模块，在这里暂时不采取处理。

4.3.2 ffplay 函数调用流程

ffplay 播放媒体文件所需要的主要函数流程如下：首先调用位于 ffplay.c 第 3099 行 main 函数，然后调用 stream_open 函数打开流。stream_open 作用是分配全局数据结构，初始化相关参数和启动文件解码线程。其中解码线程 decode_thread 最为重要。在 decode_thread 中，函数从视频队列 PacketQueue 中不停的取得视频包，然后调用 decode 函数进行解码，，在适当延时后做颜色空间转化并调用 SDL 库显示出来。

在 decode_thread 函数中，对视频文件进行分析，如果是视频文件，调用 stream_component_open 打开该视频流并打开 video_thread 进程。video_thread 进程

主要完成分配解码帧缓存和 SDL 显示缓存后进入解码循环(从队列中取数据帧，解码，计算时钟，显示)，释放视频数据帧/数据包缓存。该进程通过调用 get_video_frame 函数中打开一个包获取包中的视频帧，然后调用 avcodec_decode_video2（在 0.5.3 以前是 avcodec_decode_video 函数）完成实质性的将视频帧解码成 YUV 或 RGB 格式数据文件的工作。avcodec_decode_video2 就是是解密工作的切入点。这个流程函数调用关系如下图所示：

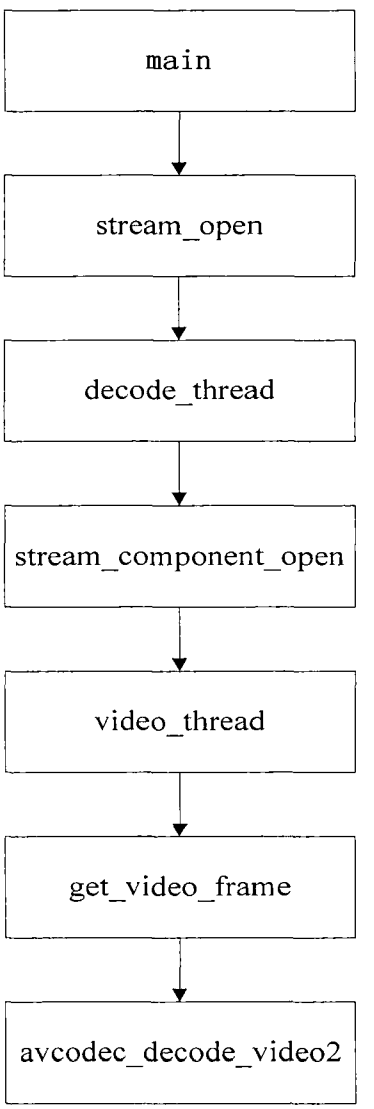

图 4.3 ffplay 函数调用关系流程图

4.3.3 具体解密算法实现

在转码加密模块，关键视频帧被异或处理得到了加密后的视频文件。由于异或

算法的自反性，在播放解密模块，只需要对关键视频帧再次进行异或运算，就可以得到加密前正常的视频帧。

在进行解密前，还应该进行判断该视频是否为加密视频，如果是的话，则进行解密处理，如果不是，则不做任何处理。这样就可以保证播放器可以播放未经加密的视频。

本系统采用的异或算法与加密段完全一致，本质是将 data 变量转换为字符串，然后将其与另外一个字符串（即 key）进行按位异或运算，用伪代码描述如下;

str 表示将要被加密的字符串，key 表示密钥。sPos 与 kPos 表示字符串 str 与密钥 key 的字符成员所在位置，初始值为 0，即都从字符串的第一位开始。

（1）在 sPos 小于字符串长度的情况下

IF kPos 大于等于密钥长度，则置 kPos 为 0

从 0 位开始对 str 与 key 进行按位以后运算

sPos 与 kPos 加一，嵌套调用 xorEncrypt 函数

（2 ）IF sPos 大于字符串长度（按位异或运算已经完成）

返回经过处理后的 str 值

解密关键代码如下：

```
static char *xorEncrypt(char *str, char *key, int sPos, int kPos)
{
    if (sPos < strlen(str))
    {
        if (kPos >= strlen(key))
        {
            kPos = 0;
        }
        str[sPos] = str[sPos] ^ key[kPos];
        return xorEncrypt(str, key, ++sPos, ++kPos);
    }
    else
    {
        return str;
    }
}
```

该函数定义与转码加密模块完全一致。

具体解密函数应位于 avcodec_decode_video2 函数之前，如下

```
is->video_st->codec->reordered_opaque= pkt->pts;
if(pkt.flags==1&&pkt.encrypt==1)//判断是否为关键帧和是否已经加密
{
 pkt.data=(uint8_t *)xorEncrypt((char *)(pkt.data),"asdfghjk",0,0);//异或运算
}
len1 = avcodec_decode_video2(is->video_st->codec,frame, &got_picture, pkt);
```

在解密端，异或算法的具体实现和加密段完全一致。在进行解密之前，首先判断该帧是否为关键帧，同时判断是否经过加密处理，只有在两种情况都符合的下，方对其采取解密处理。

按照上述方式对 ffplay.c 文件进行修改后，进行正常编译安装，所得到的 ffmpeg 系统就可以对加密视频文件进行播放了。

4.4 转码加密模块图形用户界面的实现

本节将详细介绍基于 Qt SDK（Linux 版）转码加密模块图形用户界面的具体实现过程。

4.4.1 使用 Qt designer 视图设计用户界面布局

首先打开 Qt creator，新建一个基于“Qt Gui Application”模板的项目，将其命名，其他选项均按照默认值设置即可。这个项目有四个文件，分别是“mainwindow.ui"、“mainwindow.h"、"mainwindow.cpp"和“main.cpp"。其中，“mainwindow.ui"是可编译的图形外观布局界面，支持拖动控件等方式对布局进行设计。“mainwindow.h”定义了项目需要的外部库和函数。"mainwindow.cpp"是项目的主题部分，是项目函数的具体实现。"main.cpp"由系统自动生成。

点击“mainwindow.ui"文件，进入编辑图形外观布局界面。默认的布局含有除菜单栏、工具栏和状态栏，在本项目中，将这三个控件删除。

（1）输入文件选择区

这个区域用于添加输入文件，采用的方式是按照系统目录树添加。

从左侧控件栏中“buttons”选项下选中一个“push button”按钮，拖动至空白画布，双击，命名为“输入文件”。从“Inputs”列表中添加一个“Line Edit”控件，拖动至空白画布“输入文件”按钮右侧，用于显示输入文件保存位置。默认情况下，此控件是交互式的，即允许用户输入字符，在这里，只需要选择文件目录，所以将其值改为“disable”。

（2）特定格式选择区

在这个区域，用户可以选择“MP4”、“手机”或“暂无”选项，对应的输出文件分别是 MP4、3GP 和自由选择格式。

从左侧控件栏中“buttons”选项下选中三个“radio button”按钮，拖动至空白画布，每个按钮代表一种特定格式。分别双击这三个按钮，分别修改其标签文本为“MP4”、“手机”和“暂无”。这三个按钮为单选按钮，用户每次只能选择一个。

（3）参数选择与后台输出区

这个区域用于选择输出视音频的参数，采用的形式是下拉列表。此处还可以显示转码过程中 FFmpeg 后台运行情况。为使界面看起来简洁美观，此处采用切换选项卡的方式（类似于浏览器的标签选项卡），来分别显示参数选择和后台输出区域。

从控件栏中的 Containers 列表把选项卡小部件拖到画布上，选中之后，点击属性列表中的“currentTabText”属性，以便修改每个选项卡上显示的文本。我们选择了“参数选择”和“后台输出”。

在“参数选择”选项卡上，添加 5 个标签与组合框，其中 3 个对应视频参数，2 个对应音频参数。通过组合框的下拉选项中，用户可以选择视频的分辨率、帧率和比特率，以及音频的采样率和压缩比特率。

在“后台输出”选项卡上，添加一个 TextBrowser 控件，这个控件用来显示 FFmpeg 的后台输出。

（4）输出文件区

这个区域与输入文件区域类似，都是“push button”按钮和“Line Edit”的组合。不同的是，在输出文件区域中，“Line Edit”的值是“enable”，即允许输入字符。在参数选择区选择“暂无”的情况下，用户可以在这里输入字符，来得到需要的视频格式。

此外，还需要一个按钮，此按钮类似于开关，点击后系统即开始转码工作。该按钮双击后命名为“开始”。

至此，图形外观布局界面设计完成，完整的用户界面如下图 4.4 所示：

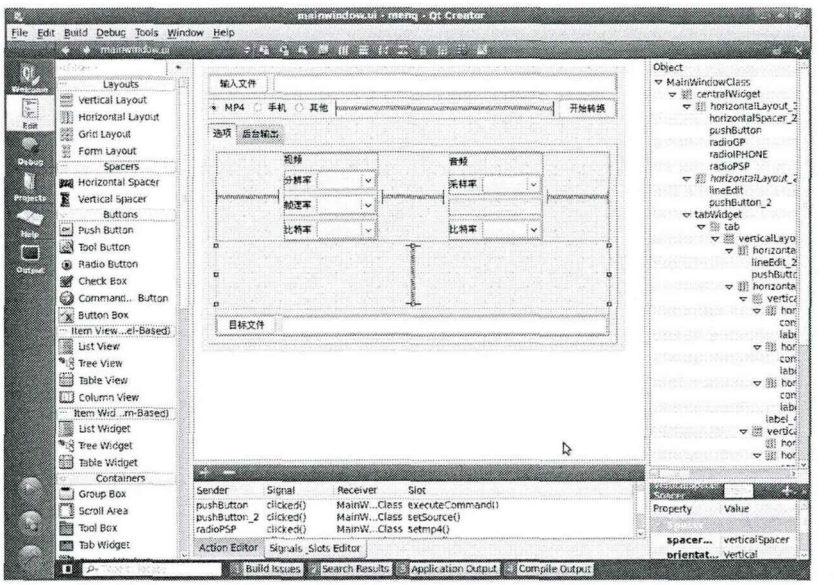

图 4.4　图形外观布局界面设计

4.4.2 控件对应函数功能设计

（1）信号与槽

在 Qt 中，控件和对应函数进行通信是基于 Qt 独有的信号与槽机制产生的。在一般的图形用户编程中，控件与函数的通信是采用回调函数的方式进行，这些回调函数通常是一个指向某个函数的指针[44]。在 QT 中，信号和槽机制取代了这些复杂的函数指针，使得编写这些通信程序更为简洁明了，信号和槽能携带任意数量和任意类型的参数，其类型是完全安全的[45]。

当控件改变其状态时（点击或输入等），信号就由该控件发射 (emit) 出去。槽用于接收信号，是普通的对象成员函数。在 Qt 中，信号用关键字“signals”定义，槽用关键字“slots”定义[46-47]。

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。该函数的定义如下：

```
bool QObject::connect ( const QObject * sender, const char * signal,
        const QObject * receiver, const char * member )
```

（2）具体功能实现

进入 Signals/Slots 编辑模式，新建 6 个槽，分别是：

executeCommand()、setSource()、setDestination()、setMP4()、setMobile() 和 setDefault ()。这 6 个槽分别对应“开始”、“输入文件”、“输出文件”、“MP4”、“手

机”和“默认”按钮的 clicked 信号。

➢ **mainwindow.h 文件**

打开 mainwindow.h 文件，添加对槽的属性与功能定义，如下表 4.1 所示：

表 4.1 槽的属性与功能定义

编号	定义	含义
1	private slots:	私有槽
2	void executeCommand();	可执行文件槽
3	void setSource();	源文件槽
4	void setDestination();	目的文件槽
5	void setMP4();	MP4 格式槽
6	void setMobile();	手机格式槽
7	void setDefault();	默认格式槽
8	void outputCommand();	输出命令槽

其中，outputCommand 用来显示 FFmpeg 后台输出。

还应添加程序中需要用到的几个文件头：

```
#include <QByteArray>
#include <QProcess>
#include <QTextBrowser>
#include <QDesktopServices>
#include <QFileDialog>
#include <QComboBox>
```

其中，QbyteArray 和 Qprocess 用于在程序中运行 FFmpeg 并抓取其命令行输入参数和输出文本。QtextBrowser 用来显示文本输出。QdesktopServices 和 QfileDialog 用来以目录树的形式寻找输入文件和定位输出文件。QcomboBox 则和参数选择区中的下拉菜单有关。

另外还需定义一个外部程序变量：QProcess FFmpeg

➢ **mainwindow.cpp**

mainwindow.cpp 里面包含了上诉槽的具体实现。部分关键代码如下：

获取 FFmpeg 输入命令：

```
void MainWindow::executeCommand()
{
    QStringList args;
    args<<"-i";
```

```
    args<<ui->lineEdit->text();//输入文件
    args<<"-s";
    args<<ui->comboResolution->currentText();//设置视频分辨率
    args << "-r";
    args << ui->comboFramerate->currentText(); //设置视频帧速率
    args << "-b";
    args << ui->comboBitrate->currentText(); //设置视频比特率
    args << "-ar";
    args << ui->comboSamplerate->currentText(); //设置音频采样频率
    args << "-ab";
    args << ui->comboBitrate->currentText(); //设置音频压缩比特率
    args << ui->lineEdit_2->text();     //输出文件
    commandProcess.start("ffmpeg", args); //执行外部程序 FFmpeg
}
```

设置输入文件：

```
void MainWindow::setSource()
{
    QString file = QFileDialog::getOpenFileName(this, tr("选择输入文件"),
    QDesktopServices::storageLocation(QDesktopServices::MoviesLocation));
    ui->lineEdit->setText(file);
}
```

设置输出文件：

```
void MainWindow::setDestination()
{
    QString file = QFileDialog::getSaveFileName (this, tr("保存输出文件"),
    QDesktopServices::storageLocation(QDesktopServices::MoviesLocation));
    ui->lineEdit_2->setText(file);
}
```

设置 MP4 格式参数：

```
void MainWindow::setMP4()
{
    ui->comboResolution->clear();//清空分辨率
    ui->comboFramerate->clear();//清空帧速率
    ui->comboBitrate->clear();//清空比特率
```

```
    ui->comboSamplerate->clear();//清空采样频率
    ui->comboAbitrate->clear(); /清空压缩比特率
    ui->comboResolution->addItem("240x320");//分辨率
    ui->comboResolution->addItem("120x160");//分辨率，这是一个下拉的对话
框
    ui->comboFramerate->addItem("24");//帧速率
    ui->comboBitrate->addItem("1000");//比特率
    ui->comboSamplerate->addItem("2000");///采样频率
    ui->comboAbitrate->addItem("65000");//压缩比特率
    ui->lineEdit_2->setText("ccnu.mp4");//格式为 MP4
}
```

其中，参数可以根据实际情况多列出几种。

设置手机格式：

```
void MainWindow::setMobile()
{
    ui->comboResolution->clear();//清空分辨率
    ui->comboFramerate->clear();//清空帧速率
    ui->comboBitrate->clear();//清空比特率
    ui->comboSamplerate->clear();//清空采样频率
    ui->comboAbitrate->clear();//清空压缩比特率
    ui->comboResolution->addItem("480x272");//分辨率
    ui->comboFramerate->addItem("24");//帧速率
    ui->comboBitrate->addItem("1000");//比特率
    ui->comboSamplerate->addItem("4800");//采样频率
    ui->comboAbitrate->addItem("128k");//压缩比特率
    ui->lineEdit_2->setText("ccnu.3gp");//格式为 3gp
}
```

设置默认格式：

```
void MainWindow::setDefault()
{
    ui->comboResolution->clear();//清空分辨率
    ui->comboFramerate->clear();//清空帧速率
    ui->comboBitrate->clear();//清空清空比特率
```

ui->comboSamplerate->clear();//清空采样频率

ui->comboAbitrate->clear();////清空压缩比特率，默认格式即输出文件和输入文件参数保持一致

}

将后台输出选项卡信号与 outputCommand 建立关系：

connect(&commandProcess,SIGNAL(readyReadStandardOutput()),this,SLOT(outputCommand()));

connect(&commandProcess,SIGNAL(readyReadStandardError()),this,SLOT(outputCommand()));

显示 FFmpeg 后台输出：

void MainWindow::outputCommand()

{

QByteArray cmdoutput = commandProcess.readAllStandardOutput();//命令标准输出

QString txtoutput = cmdoutput; //转化为字符串

ui->textBrowser->append(txtoutput); //文本输出

cmdoutput = commandProcess.readAllStandardError();//错误提示输出

txtoutput = cmdoutput;//转化为字符串

ui->textBrowser->append(txtoutput);//文本输出

}

FFmpeg 后台输出有两种：来自命令的标准输出和发生错误时的错误提示输出。Qt 将其分别抓取，然后转化为字符串，最后让其以文本的形式呈现出来。

4.4.3 程序最终界面

整个程序完成后，对程序进行调试，选择输入文件，确定好相关参数和输出文件位置后，就可以开始转换并加密了，程序主界面如下图 4.5 所示：

图 4.5　最终界面设计图

该界面简洁美观，便于用户使用 FFmpeg 进行转码工作，程序独立于 FFmpeg 程序之外，不会对其的源代码任何干扰。

4.5 系统测试

对该系统进行测试包括转码加密和视频播放解密两方面。

在转码加密方面，采用将 asf 格式转换为 wmv 格式进行测试。用户界面如下图 4.6 所示：

图 4.6 用户界面

后台输出界面如下图 4.7 所示：

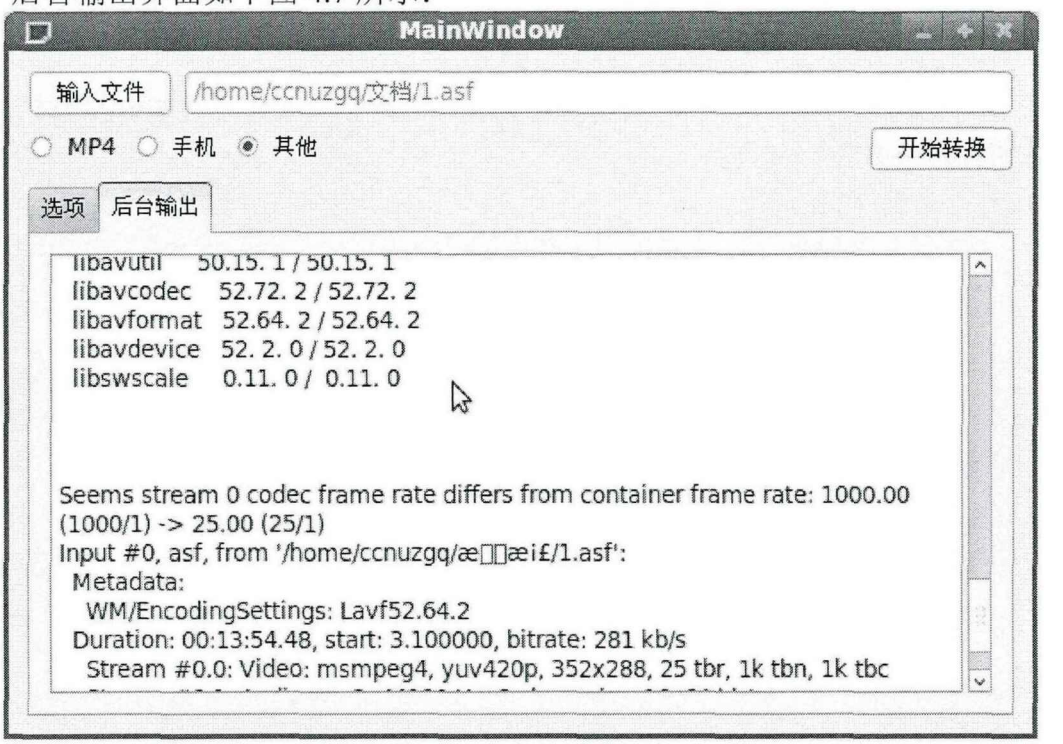

图 4.7 后台输出界面

在转换完成后，用普通播放器（暴风影音）对转换后的 ccnu.wmv 文件进行播放，出现花屏。如图 4.8 所示：

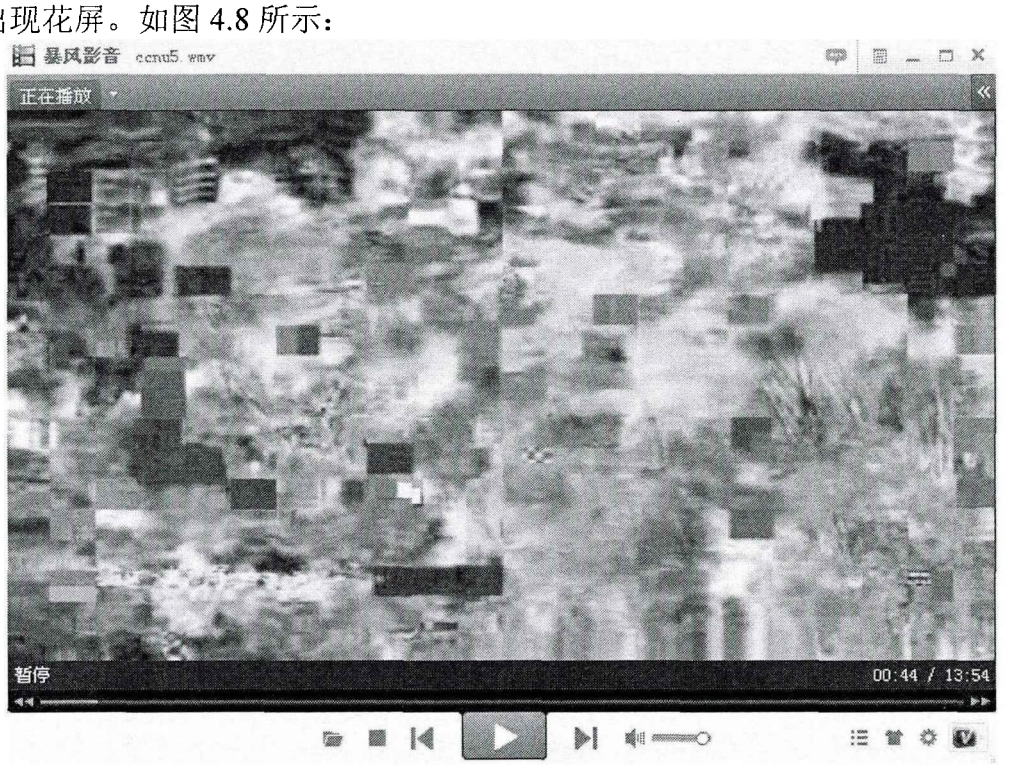

图 4.8 加密后文件播放效果

当文件用具有解密功能的播放器进行播放时，可以正常显示图像。如图 4.9 所示：

图 4.9 用解密播放器进行播放效果图

4.6 本章小结

本章首先介绍了 FFmpeg 转码加密模块的流程、函数调用关系与具体加密算法的实现和 ffplay 视频解密播放模块的具体实现，随后介绍了 Qt 应用程序框架的安装过程，接下来介绍了 FFmpeg 转码图形用户界面的具体实现方法，最后对整个系统进行了测试。

5 总结与展望

5.1 全文总结

本文首先介绍了与研究相关的理论与技术基础，然后介绍了 FFmpeg 在 Linux 环境下的配置安装过程、FFmpeg 的相关参数意义与基本使用语法，随后又分图形界面模块、转码加密模块和播放解密模块对整个系统进行了设计，最后分别详细阐述了这三个模块的具体实现。具体来说，本文主要做了以下几方面的工作：

（1）分析了国内外对于 FFmpeg 的研究现状，并根据其存在的不足提出了该论文研究的主要内容。指出了国内外对 FFmpeg 的研究很少涉及到数字版权保护领域，引出了本论文特色与创新之处。

（2）针对问题密切相关的理论与技术基础做了较好的综述，分别介绍了 Linux 系统的特点，GUI 图形界面的定义和优点，DRM 数字版权保护系统主要采取的技术和 Qt 用户界面应用框架的特性与模块，为后面系统的设计与实现打下了坚实的基础。

（3）阐述了 FFmpeg 在 Linux 平台上的配置安装过程，包括 Linux 环境的搭建方式、外部库的安装和 FFmpeg 的编译与安装，介绍了 FFmpeg 的相关选项参数意义和具体使用方法。

（4）分析了视频文件的相关背景知识，然后对系统进行了需求分析，接下来对整个系统按照转码用户界面模块，转码加密模块和播放解码模块来分别进行了设计。

（5）提出了 FFmpeg 转码图形用户界面的具体实现方法，随后介绍了 FFmpeg 转码加密模块的流程、函数调用关系与具体加密算法的实现和 ffplay 视频解密播放模块的具体实现，最后对整个系统进行了转码加密和播放解密测试。。

5.2 展望

FFmpeg 是一个非常庞大的系统，几乎实现了世界上所有的多媒体编码解码。Ffmpeg 源代码异常复杂，阅读理解难度较大。目前关于 FFmpeg 的参考资料较少，仅大部分停留在源代码的编译步骤，和原理性的介绍方面。所以本论文所设计的基于 FFmpeg 视频版权保护系统仅仅是一个阶段性的成果，不足之处还有很多，还需要进一步的研究与实现。下一步的工作主要考虑以下几个方面：

（1） 完善不同格式间转换与加密的功能

由于本设计中的视频加密解密是以 AVPacket 为基础进行设计，所以目前只能实现 msmpeg 系列编解码标准的视频加密解密功能，主要包括 asf，wmv 等视频文件。在后续研究中，应改变研究切入点，不断增加可以支持的视频格式。

（2）改进加密解密算法

本设计中的加密解密算法是采用的最为简单的异或运算，异或运算加密复杂度不高，不利于系统的实际应用。下一步可以考虑采用更加复杂的加密算法对视频文件进行加密，以提高系统的抗干扰性。

（3）播放器用户界面设计

本设计播放器是基于 FFmpeg 项目中自带的 ffplay 播放器进行开发的，默认的是采用的命令行的方式进行操作。下一步可以考虑为该播放器设计图形化用户界面，方便使用者使用。

参考文献

[1] 杨贵明.从中国广播网的应用看流媒体技术的发展[J]. 广播与电视技术 2004(7):50-52.

[2] 任严,韩臻,刘丽.基于 FFMPEG 的视频转换与发布系统[J].计算机工程与设计.2007(20):4962-4967.

[3] 马洪堂.基于 FFmpeg 的视频转换系统的模块研究[J]，电脑知识与技术.2009(5):3545-3546.

[4] 单海涛.基于 ffmpeg 的高清数字电影软件编码系统的设计[J]，信息技术，2007(1):96-99.

[5] 吴张顺.基于 FFmpeg 的视频编码存储研究与实现[J]，杭州电子科技大学学报，2006(3):30-34.

[6] 田虎.DRM 系统的组成及设计[J],电视技术,2009(32).86-88.

[7] 程春玲.实现 DRM 系统的一种新方案[J]，计算机技术与发展.2009(7)：166-168

[8] Abhishek Udupa, Sreepathi Pai. Optimizing the ffmpeg Library for the Cell BE[C], CA E0243 Aug 2006 Course Project Report.

[9] Heikki Orsila .Trust Issues in Open Source Software Development[C]. WUP '09 Proceedings of the Warm Up Workshop for ACM/IEEE ICSE.2010.

[10] Memon N,Wong Ping Wah. A Buyer-Seller WatermarkingProtocol[J].IEEE Tranctions on Image Processing,2001,10(4):643-649.

[11] 郝振省.2008 中国数字版权保护研究报告[M].中国书籍出版社，2008 年.

[12] 赖宏萍.Linux 的嵌入式研究应用[D].浙江大学 2005 年硕士毕业论文.

[13] Linux 的优点[EB/OL]，http://www.51cto.com/html/2005/0907/2484.htm.

[14] 陈　功, 黄祥林,沈兰荪. 视频转码技术[J].测控技术.2003(5):36-39.

[15] Dutta,R.Mukhopadhyay,S. Dowling, T. Key management in multi-distributor based DRM system with mobile clients using IBE. Applications of Digital Information and Web Technologies, 2009. ICADIWT '09. Aug,2009.:597

[16] 杨成，杨义先.信息安全与数字版权保护[J].计算机安全.2004 年(1):32-36.

[17] S.R.Digital right management[C].IEEEPOTENTIALS.2006.20(4):31-34

[18] 张玲峰.数字水印技术在版权保护中的应用研究[D].湖南大学工程硕士论文.

[19] Kalker, T. DRM Interoperability. Computational Intelligence and Security[J].2007(9).

[20] GUI[EB/OL].http://baike.baidu.com/view/25309.htm#sub25309.

[21] Qt,一个跨平台的应用程序和 UI 开发框架[EB/OL]，http://qt.nokia.com/title-cn/.
[22] 刘治.基于 RFB 协议跨平台网络远程监控技术的研究与实现[D]，北京化工大学 2009 年硕士毕业论文.
[23] 中国软件技术有限公司著.Qt 程序设计[M].北京：清华大学出版社，2002.
[24] 使用 Qt 和 OpenGL® 创建跨平台可视化 UI[R].诺基亚公司
[25] Xiaoming Sun, Bin Qi, Xiude Zhang. Qt-based application of virtual model making system in agricultural machinery[C]. Computer Science and Education (ICCSE), 2010 5th International Conference. 2010(4):1014 – 1017.
[26] FFmpeg 工程组[EB/OL]. http://www.ffmpeg.com.cn/index.php.
[27] ffmpeg[EB/OL].http://ffmpeg.org/.
[28] Tong Lai Yu Turner, D. Stover, D. Concepcion, A. Incorporating video in platform-independent video games using open-source software[J]. Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference.2010, (9):466
[29] 刘巍巍.Linux 操作系统配置服务器的方法及实现[J],中国科技信息，2010 (4):88-90.
[30] 在 windows 下编译 FFMPEG[EB/OL].http://hi.baidu.com/xulina809/blog/
[31] Linux 下软件安装详解[EB/OL]
http://www.pconline.com.cn/pcjob/system/Linux/others/0507/672129.html
[32] ffmpeg 编译及其使用[EB/OL]
http://www.360doc.com/content/10/0428/15/474846_25293076.shtml
[33] 多媒体视屏技术相关术语[J].软件工程师 2004(11):63
[34] 杜旭峰, 视频压缩编码技术研究[D].北京邮电大学 2005 年硕士毕业论文.
[35] FFPLAY 的原理[EB/OL].
http://blog.csdn.net/shenbin1430/archive/2009/06/23/4291893.aspx.
[36] 常见视频文件的编码方式和封装格式[EB/OL].
http://blog.csdn.net/wxzking/archive/2010/07/19/5746641.aspx.
[37] 黄勇.视频压缩技术探讨[J].科技信息学术版.2008(20):162-163
[38] 维基百科 MPEG [EB/OL] .http://zh.wikipedia.org/wiki/MPEG
[39] 程文青,邓婉婷,刘清堂.数字媒体版权管理平台研究与实现[J].计算机应用与软件,2007,24(7):27-29.
[40] Suramya Tomar .Converting video formats with FFmpeg[M]. Specialized Systems Consultants, Inc. 2219 NW Market Street Seattle, WA USA.2006
[41] 谭昌盛，曾碧，王国伟. 基于 3G 视频传输的数据包加密及其实现技术[J]. 现

代计算机.2010(11):99-101.

[42] FFMpeg 对视频文件进行解码的大致流程[EB/OL].
http://arm9.org.ru/viewthread.php?tid=535.

[43] 杨书良.FFMPEG/FFPLAY 源码剖析[R].

[44] Nokia 公司.Qt 参考文档[EB/OL].http://doc.trolltech.com/4.6/index.htm.

[45] 肖世杰.嵌入式图形系统若干关键技术研究[D].华中科技大学 2004 年硕士论文.

[46] Jasmin Blanchette,Mark Summerfield.C++GUI Programming with Qt3[M].Pretice Hall Ptr.2001.

[47] Jasmin Blanchette,Mark Summerfield.C++GUI Qt4 编程（第二版）[M].北京：电子工业出版社，2008

攻读学位期间发表的学术论文

[1] Zhu Xiao-liang,Zhang Guo-Qing,Song Xiao-juan. Research on MDC transmission over Wireless Mesh Network. Future Computer and Communication (ICFCC). Wuhan. 2010,2.pages :V2-361 - V2-365

[2]马晓娟,刘清堂,张国庆.教育技术学专业的 SWOT 分析[J].中国教育技术装备,2011,2(6):6~10.

致 谢

斗转星移，时光荏苒，转眼间，已经到了临近毕业的时刻。回首在华师的求学经历，对那些引导我、帮助我、激励我的人，我心中充满了感激。

首先要感谢导师刘清堂教授和朱晓亮副教授。在学习科研上，刘老师一丝不苟，严格要求。在日常生活中，他关心备至，和蔼可亲。无论是在科研方面还是在职业规划方面，刘老师的指导均高屋建瓴，发人深思。刘老师学术造诣精湛，品德高风亮节，研究生三年，我从刘老师身上学到了许多做学问和做人的道理，这些道理必将令我受益一生。在这里，我还要感谢朱晓亮老师。朱老师宽厚仁慈，学识渊博。是我平时学习和科研的直接指导老师。无论是在研一研二的学习科研中，还是在研三的就业和书写毕业论文的过程中，朱老师均给予了我耐心细致的指导。研究生生活刚开始时，自己在科研上遇到了一些问题，正是朱老师不厌其烦耐心细致的指导才使我克服了畏难情绪，顺利地完成了科研任务。在这里，我要感谢两位老师对我的帮助，谨记老师教诲，在以后的工作中勤奋踏实工作，为师长争光，为母校争光。

我要感谢信息技术系所有老师。从 2004 年 9 月开始至今，我在华中师大信息技术系已经度过了 7 年的时光。七年前刚刚踏入华师，进入信息技术系学习时的情景仍历历在目。信息技术系老师精于学术，为人正直，为我学习和生活上提供了许许多多的帮助，我以在华师信技系度过我的大学生活为荣！

我还要感谢周围的同学们和宿舍的室友。谢谢他们一直以来对我的支持与帮助。祝他们以后工作顺利，生活幸福！

我要感谢父母对我的养育之恩。求学十余载，全靠父母在背后默默支持。对于他们，任何感激的言语都是苍白的，唯有以后多尽孝道，让他们安心。

最后，我要感谢参与我论文评审和答辩的各位老师，他们给了我一个审视几年来学习成果的机会，他们对我的帮助是一笔无价的财富。

在论文即将完成之际，感慨万千，一切即将结束，一切又将开始。踏上新的征程，我会不懈努力，继续奋斗，来报答所有关心、支持、鼓励和帮助我的人！

张国庆

2011 年 5 月于桂子山