

**本科学生毕业设计（论文）**

题 目： 音视频播放系统的设计与实现

学生专业： 软件工程

学生班级： 软工19-4班

学生姓名： 荆健威

学 号： 190702940915

指导教师： 杜娟

辅助教师：

2023年 5 月 5 日

摘 要

本文介绍了一款音视频播放系统的设计与实现过程，并对音频解码、重采样以及视频解码技术进行了研究。该系统使用C++作为开发语言，使用qt作为窗体界面，通过FFmpeg库函数实现了对音视频文件的解码、解封装。

本系统首先将封装好的音视频文件解封装，再从解封装得到的音视频流中获取数据包，并将数据包发送到相应的解码器。对于视频数据包，系统将其发送至视频解码器，将压缩后的视频数据（例如H.264、AVC等格式）转换为可渲染的图像序列；对于音频数据包，系统将其发送到音频解码器，将压缩后的音频文件（如MP3、AAC等格式）还原成无损音频数据，最后将这些数据进行渲染和同步，通过音视频播放系统播放出来。

该播放器能够实现一般的播放器的基本功能，同时支持多线程播放，基本功能包括：音视频文件的打开、关闭、播放、暂停播放、停止播放、视频进度控制、音量的调整等。

**关键词**：音视频播放系统；FFmpeg；qt；解码；解复用

# 

# Abstract

This paper introduces the design and implementation process of an audio and video player, and investigates audio decoding, resampling and video decoding techniques. The system uses C++ as the development language, qt as the form interface, and the decoding and decapsulation of audio and video files are realized through FFmpeg the library function.

The system firstly decapsulates the encapsulated audio and video files, then gets the packets from the audio and video streams obtained from the decapsulation and sends the packets to the corresponding decoder. For video packets, the system sends them to the video decoder to convert the compressed video data (e.g. H.264, AVC and other formats) into renderable image sequences; for audio packets, the system sends them to the audio decoder to reduce the compressed audio files (e.g. MP3, AAC and other formats) into lossless audio data, and finally renders and synchronizes these data to be played out through the The audio and video player is played out.

The player can achieve the basic functions of a general player, while supporting multi-threaded playback. The basic functions include: opening, closing, playing, pausing playback, stopping playback of audio and video files, video progress control, volume adjustment, etc.

**Key words：**Audio and video player；FFmpeg；qt；decode；Demultiplexing

**目 录**

**[第1章 概 述 1](#_Toc6197)**

[1.1系统开发背景 1](#_Toc1578)

[1.2系统开发的目的和意义 2](#_Toc12375)

[1.3 系统功能简介 3](#_Toc16158)

**[第2章 相关开发工具及环境介绍 5](#_Toc28306)**

[2.1 相关开发工具概述 5](#_Toc462)

[2.2 相关开发技术概述 5](#_Toc9097)

[2.3 音视频格式介绍 6](#_Toc31664)

**[第3章 可行性研究及需求分析 8](#_Toc5020)**

[3.1 可行性研究 8](#_Toc27360)

[3.2系统的需求分析 10](#_Toc18282)

**[第4章 音视频播放系统总体设计 13](#_Toc3196)**

[4.1音视频播放系统架构总体设计 13](#_Toc26656)

[4.2 音视频播放系统功能的总体设计 14](#_Toc25558)

[4.3 音视频文件到播放总体设计 15](#_Toc22336)

**[第5章 音视频播放系统的详细设计与实现 22](#_Toc21010)**

[5.1音视频主线程模块的详细设计与实现 22](#_Toc2138)

[5.2音频线程模块的详细设计与实现 25](#_Toc16250)

[5.3视频线程模块的详细设计与实现 28](#_Toc24997)

[5.4图片绘制模块的详细设计与实现 30](#_Toc25646)

[5.5锁和条件变量的详细设计与实现 31](#_Toc8801)

[5.6音视频文件seek的详细设计与实现 33](#_Toc21393)

[5.7音视频同步详细设计与实现 33](#_Toc1305)

**[第6章 音视频编解码技术实现 35](#_Toc18404)**

[6.1视频编码技术实现 35](#_Toc17739)

[6.2视频解码技术实现 36](#_Toc22506)

[6.3音频编码技术实现 37](#_Toc11009)

[6.4音频解码技术实现 38](#_Toc20114)

**[第7章 系统测试 39](#_Toc5201)**

[7.1系统测试分析 39](#_Toc23791)

[7.2系统测试方法 39](#_Toc29679)

[7.3系统测试用例 40](#_Toc24850)

[7.4测试分析总结 41](#_Toc13615)

**[结论 44](#_Toc4218)**

**[参考文献 45](#_Toc14628)**

**[致谢 46](#_Toc2868)**

第1章 概 述

## 1.1系统开发背景

在当今信息社会，以多媒体为代表的信息技术和信息产业的发展和应用对人类社会产生的影响和作用愈来愈明显，愈来愈重要。多媒体的发展和应用，极大地推动了诸多工业的相互渗透和飞速发展，逐步改变了整个人类社会的工作结构和生活方式。可毫不夸张地说，多媒体产业的形成和发展，将不仅引起计算机工业的一次革命，也将影响人类社会发生一场巨大的变革[1]。

流媒体中的文本、图像、动画、音频、视频等多媒体信息源通过特殊的编码、压缩和分割，被分割成压缩的数据包，然后由视频服务器终端计算机实时传输。

然后由视频服务器终端计算机实时传输这些数据包。在播放的同时，流媒体文件服务器的其余部分支持在后台继续下载。因为这个过程是动态的，多媒体在网络上的传输就像水一样不断延伸。水不断延伸，所以形象地称为流媒体[2]。

由于多媒体技术使计算机具有综合处理声音、文字、图像和视频信息的能力，它以形象丰富的声、文、图等信息和方便的交互性，极大地改善了人机界面，改变了使用计算机的方式，从而为计算机进入人类生活和生产的各个领域打开了大门。同时，多媒体计算机技术是信息高速公路架构的重要组成部分，是我国国民经济信息化的核心技术。多媒体系统需要将不同的媒体数据表示成统一的结构码流，然后对其进行变换、重组和分析处理，以进行进一步的存储、传送、输出和交互控制[3]。

多媒体编码技术可以将音频、视频等各种形式的媒体内容压缩成数字数据，以便在网络上传输和存储。常见的编码标准包括MPEG-2、H.264、VP9等。随着编码技术的不断改进和优化，数字媒体的质量和效率逐渐提高，能够更好地满足用户需求[4]。多媒体编码、网络传输和用户界面等方面的创新对媒体技术的发展起到了至关重要的作用。这些技术的不断改进和优化推动了数字媒体技术的进步，并为人们提供了更丰富、更便捷的数字娱乐服务[5]。

流媒体服务已经成为数字娱乐产业变革的重要驱动力[6]，流媒体服务指的是通过互联网将音乐、电影、电视节目等数字媒体内容传输到消费者设备上的一种服务。随着数字技术的发展，流媒体服务已经成为数字娱乐产业变革的重要驱动力。

流媒体传输具有以下特点：流媒体传输需要实时性，流媒体传输需要保证数据能够在短时间内到达用户端，否则会影响用户观看体验。对带宽和网络质量的依赖性较高,流媒体传输需要占用相当的带宽，而网络质量差的情况下容易出现卡顿、卡死等问题。需要支持多种媒体格式,流媒体传输需要支持音频、视频等多种媒体格式，因此需要解决兼容性问题[7]。

在保证流媒体传输安全的时，应该采用一些相关的加密技术如SSL、TLS、IPSec等，这些技术都可以用于流媒体传输的加密保护。但是，这些技术也存在一些缺陷和限制, 方案复杂：现有的加密传输技术方案通常比较复杂，需要涉及多个技术环节，并且需要进行复杂的配置和管理。影响传输效率：加密传输技术会增加数据包大小和传输延迟，从而影响传输效率。安全威胁：一些加密传输技术在实际应用中存在安全漏洞和风险，例如容易受到中间人攻击、数据篡改等[8]。

随着多媒体技术的发展，音频、视频等媒体资源对于人们的日常工作、学习、生产、生活的作用越来越显著，音视频播放软件已经成为必不可少的媒体播放工具。在我们的生活中，处处可见媒体播放软件的应用：在校园里，教师为学生们播放丰富的学习资源；在公司、企业内部进行宣讲、汇报；人们日常生活中的休闲娱乐等都需要播放器软件。随着直播行业等线上互动方式的迅速发展，直播间、视频会议等活动也需要基本的播放软件技术支持。

播放器软件就是多媒体资源的播放工具，是能够读取视频或音频文件的软件。目前，多媒体播放软件技术已经非常成熟，专业的播放工具包括KMPlayer、绚彩魅影、暴风影音播放器、影音风暴、超级兔子快乐影音、Realplayer播放器、windows media player、Quicktime等等。

## 1.2系统开发的目的和意义

音频、视频播放器软件是音视频资源展现的必备工具，能够为人们欣赏多媒体资源、控制播放方式、播放效果等提供有效的手段。系统开发的主要目的和意义如下：

1、实现音视频解码、使得多媒体资源顺利播放

要实现音视频解码并使多媒体资源顺利播放，需要使用相应的编解码器和播放器。编解码器是用于将原始的音视频数据进行编码和解码的软件或硬件，而播放器则负责读取和解码音视频文件，并将其显示和播放出来[9]。

常见的编解码器有H.264、HEVC、VP9等，它们可以将原始的音视频数据进行压缩，减小文件大小并提高传输效率。而常见的播放器则包括VLC、Windows Media Player、QuickTime等，它们可以解码各种格式的音视频文件，并提供了播放控制、音量调节等功能。

在实现音视频解码的过程中，还需要考虑音视频的同步问题。因为音频和视频是两个不同的流，如果播放的时候没有同步，则会导致声音和画面不匹配的问题。因此，需要使用时间戳来确保音视频的同步，即在解码时，根据时间戳将音视频数据放到正确的位置上。

2、实现多媒体资源自由控制

要实现多媒体资源的自由控制，需要提供用户可以自定义的播放控制接口，包括暂停、播放、快进、快退、音量调节等功能。同时，还需要提供播放列表、循环播放、随机播放等功能，以满足用户的不同需求。

为了实现这些功能，需要在播放器中添加相应的控制逻辑和用户交互界面。例如，在播放界面上添加暂停、播放、快进、快退等按钮，并在代码中实现相应的逻辑，使得点击按钮时可以实现相应的操作。

此外，还需要考虑不同平台之间的差异，例如在移动设备上的交互方式与桌面设备不同。因此，需要针对不同的设备类型和操作系统，设计相应的交互界面和控制逻辑。

3、为当下流行的直播产业提供技术支持

直播与音视频播放的基本原理是相同的，都是将音视频数据进行解码和播放。不同之处在于，直播是实时的，需要在不断接收数据的同时进行解码和播放，并保证低延迟和高稳定性。

为了提供技术支持，需要先了解直播的整个流程。首先，直播端需要将音视频数据采集并编码，然后通过网络传输到服务端。服务端再将数据分发给多个观众端，并进行解码和播放。观众端需要根据网络情况和自身设备的性能来进行解码和播放。因为直播数据是实时传输的，所以网络的稳定性和带宽大小都会影响观众端的观看体验。如果网络带宽不足或者网络延迟较大，就可能出现卡顿、掉帧等问题，影响观众的观看体验[10]。

此外，观众端的设备性能也会影响解码和播放的效果。如果设备性能较低，就可能无法流畅地播放高清视频，或者播放过程中出现卡顿、延迟等问题。因此，观众端需要根据自身设备的性能和网络情况来选择合适的视频质量，并采取相应的优化措施，以保证观看体验。

4、为人们提供一种休闲娱乐的方式，丰富业余生活，缓解压力

当今社会正处于飞速发展的新时代，现代人在享受丰富多彩的现代生活的同时，也承受了来自于方方面面的巨大压力，例如成年人面对的工作压力、每个家庭面临的生活压力、青少年群体面对的学业压力等等。诸多压力带来了一些负面影响，使人们产生很多消极情绪，因此，无论是成年人还是青少年都需要一些缓解压力、调节生活的手段，需要劳逸结合、释放情绪。

目前，人们主要的解压方式包括网络休闲、户外活动、体育锻炼、娱乐活动等。而无论是在哪一种休闲方式下，音视频播放器都是必不可少的工具，能够为人们提供一种欣赏和控制多媒体资源的工具。健身的同时听听歌、刷刷剧；娱乐活动时播放一段小视频；甚至什么都不做只是欣赏一段轻音乐、钢琴曲，都能够使人们的身心的到极大的放松。

## 1.3 系统功能简介

本系统主要实现了对本地音视频文件的播放功能，可播放音频文件和视频文件，用户可以通过打开文件按钮去选择本地的音视频文件去进行播放，用户可以通过对音量键调整实现对声音大小的控制，可以通过静音按钮去实现静音功能，用户可以通过暂停按钮去暂停当前视频的播放，通过播放按钮去实现对视频的播放，通过停止按钮实现对视频的停止播放去播放新的视频，用户也可以去拨动进度条去实现视频的跳跃播放的功能。

音视频解码算法是指将经过压缩的音视频信号进行解码还原为原始的音视频数据的算法。常见的音视频解码算法有以下几种：

1、DCT（离散余弦变换）：DCT是一种基于数学变换的音视频编解码技术，它通过将时域或空域上的采样值转换到频域上，从而实现音视频压缩。在JPEG、MPEG等编码标准中广泛使用。

2、奈奎斯特采样定理：奈奎斯特采样定理是一种基于采样率的音视频编解码技术，它规定了采样率与信号带宽之间的关系，只要满足一定的采样率条件，就可以有效地压缩音视频信号。

3、线性预测：线性预测是一种基于预测误差的音视频编解码技术，它通过对音视频采样值之间的相对关系进行分析和建模，从而实现压缩。

4、变换编码：变换编码是一种基于变换域的音视频编解码技术，它通过对图像或音频信号进行变换，如傅里叶变换和小波变换等，然后对变换后的系数进行编码和压缩。

5、运动补偿：运动补偿是一种基于运动估计的音视频编解码技术，它通过对相邻帧之间的运动进行估计和预测，然后对预测误差进行编码和压缩，从而实现高效的音视频压缩。

实现网络直播可以通过一些流媒体服务器例如：NGINX RTMP Module，Adobe Media Server，把要直播的音视频资源通过rtmp流媒体协议进行推流，将音视频资源放入流媒体服务器中，客户端这里可以在通过rtmp，hls对服务器的音视频资源进行拉流到音视频播放器中，在通过音视频播放器的解码，渲染等功能对服务器上的视频资源进行播放，如果要实现直播也可以通过主播端的视频采集设备将主播端的视频记录下来，在通过一些处理实现主播端的美颜，瘦脸等美化功能，再将处理好的视频进行编码，之后在进行封装在通过流媒体协议将音视频推送到流媒体服务器，最后播放的步骤同上。

针对通信中音视频数据量过大、传输时间过长的问题。本研究基于用户对通信过程中音视频信号编解码需求，设计出通信音视频编解码系统，构建在Linux系统下的开源框架，应用Hi3531处理器，将AD9886A作为视频输入芯片，使用TW2984芯片用来采集音频信号。音视频编解码算法对视频信息图像进行帧间预测、运动估计、变换量化和滤波，使用离散余弦变换，避免了音视频信息逆变换中出现错配的问题。实验结果表明，本研究的编码时延为65 ms,传输、解码时延为168 ms，解码效率高[11]。

# 第2章 相关开发工具及环境介绍

## 2.1 相关开发工具概述

### 2.1.1 FFmpeg概述

本系统需要借助FFmpeg相关一些的库函数来实现对音视频的解封装，解码的实现，FFmpeg是一款音视频编解码库，是多媒体开发者的必备技能。 FFmepg提供了多种媒体格式的封装和解封装的库函数，包括音视频编码、多种协议的流媒体、多种色彩格式转换、多种采样率转换、多种码率转换等。 FFmpeg框架的基本组成包含AVFormat、AVCodec、AVFilter、AVDevice、AVUtil等模块库。

### 2.1.2 Qt概述

用qt实现了系统的界面设计。qt是一个跨平台的C＋＋图形用户界面应用程序开发框架。它可用于开发Alpine和非Alpine应用程序，如控制台工具和服务器。Qt是一个面向对象的框架，它使用了一个特殊的代码生成扩展（称为元对象编译器（Meta Object Compiler，moc）和一些宏，qt很容易扩展并允许真正的组件编程。

## 2.2 相关开发技术概述

### 2.2.1 音视频解码技术概述

音视频解码技术是本系统一个非常重要的技术，本系统可以通过解码技术将编码后的音视频流通关相关的解码解码器进行解码最后在通过一些渲染技术将音视频在播放出来。

解码就是将视频/音频压缩编码数据，解码成为非压缩的视频/音频原始数据。音频的压缩编码标准包含AAC，MP3，AC-3等等，视频的压缩编码标准则包含H.264，MPEG2，VC-1等等。解码是整个系统中最重要也是最复杂的一个环节。通过解码，压缩编码的视频数据输出成为非压缩的颜色数据，例如YUV420P，RGB等等；压缩编码的音频数据输出成为非压缩的音频抽样数据，例如PCM数据，流程图如图2-1所示。

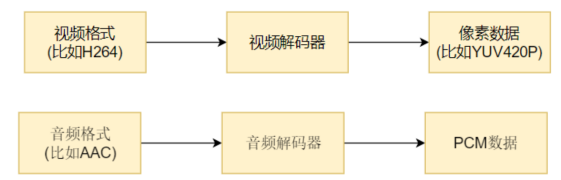


图2-1音视频解码流程图

### 2.2.2音视频解封装技术概述

本系统通过FFmpeg解复用器，去打开封装过的音视频文件，如MP4，avi等，找到对应的视频流和音频流，再通过视频流和音频流找到对应的视频压缩数据和音频压缩数据。

解封装就是将输入的封装格式的数据，分离成为音频流压缩编码数据和视频流压缩编码数据。封装格式种类很多，例如MP4，MKV，RMVB，TS，FLV，AVI等等，它的作用就是将已经压缩编码的视频数据和音频数据按照一定的格式放到一起。例如，FLV格式的数据，经过解封装操作后，输出H.264编码的视频码流和AAC编码的音频码流，流程图如2-2所示。

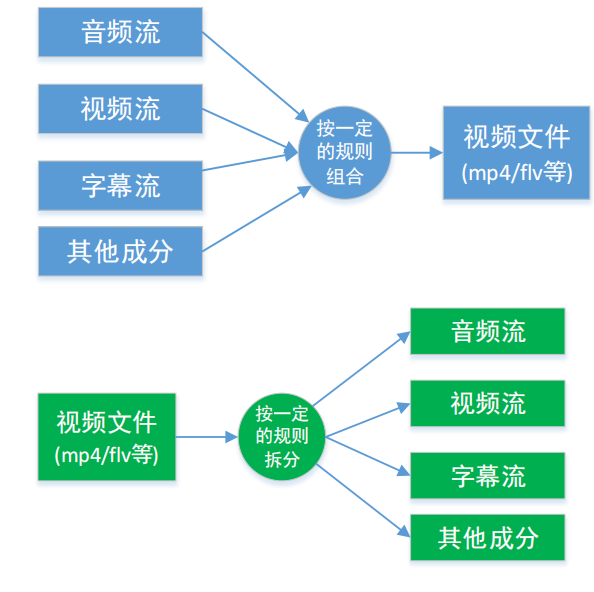


图2-2 解封装流程图

## 2.3 音视频格式介绍

常用的音视频文件是由视频和音频共同组成，其中视频文件包含MP4，MPG, MV4, 3GP，F4V，SWF，AVI，GIF，WMV，RMVB，MOV，MTS，M2T，WebM，MKV。

MP4是一种常见的视频文件格式，采用H.264/AVC视频编码和AAC音频编码。MV4是苹果公司创造的一种高清视频格式，同样采用H.264和AAC音频编码。3GP是一种功能手机时代的视频格式，采用H.263视频编码，画质较差，目前已经很少使用了。FLV是Adobe Flash软件的视频格式，现在国内所有在线视频网站都曾使用它为视频文件格式。F4V是FLV的改进版本，支持更高清视频。SWF是Adobe的一种视频格式，可以交互，用于页游、在线小游戏等。AVI采用的视频编码并不统一，但普遍相同，已经不常使用。GIF严格来说是图片格式，但也可以归其为视频的一种。WMV是微软的视频格式，采用改造过的MPEG-4视频编码和wav音频编码。RMVB是RealNetWorks公司的视频格式，现在已经几乎绝迹了。MOV是苹果的视频格式，用于QuickTime软件的影视格式。MTS是一些专业相机产生的视频格式，画质好但体积庞大。M2T是一些视频剪辑软件（如会声会影）导出的高清视频格式。WebM是谷歌的免费开放视频格式，用于HTML5视频传播。OGG是另一个免费开放的视频格式，也用于HTML5视频传播。MKV是Matroska公司为取代avi所开发的视频格式。

音频文件包括AAC，M4A，APE，FLAC，WAV，AMR，MIDI。MP3是广泛使用的音频文件格式，但有压缩和音质损失。AAC是从MP3发展而来的，具有更好的音质和更高的压缩比，现在已经是MPEG系列视频格式中的标准音频编码格式。M4A也是AAC编码的音乐文件，通常用于HTML5音乐播放。APE和FLAC都是无损音乐文件，适合音质要求较高的玩家。WAV是微软的无损音频文件，文件比较大，大多数设备录音产生的文件都是WA或AMR格式。WMA是微软的有损压缩音乐文件，音质和压缩比高于MP3，AMR主要用于移动设备的语音，压缩比较大，音质相对其他格式较差。MIDI是音乐代码，类似于乐谱，主要在音乐专业人员和乐曲爱好者中使用。

# 第3章 可行性研究及需求分析

## 3.1 可行性研究

在开始开展任务之前，通常要先进行可行性相关研究和需求分析。可行性相关研究是软件工程流程中相当重要的一个工作流程，它无需技术的设计和实现，而是对整个开发流程在各个方面开展可行性研究。

可行性研究报告的目的是为了确定一个在技术上合理、经济上可行的最优方案和最佳时机。该报告需要全面系统地分析各种因素，包括自然、社会、经济、技术等方面，并采用大量数据资料进行论证和评价。经过综合分析评价之后，可行性研究报告需要提出优缺点和建议，以便投资者能够做出明智的决策。

在可行性研究报告中，经济可行性是其中一项重要的考虑因素。该分析主要围绕着项目的投资成本、运营成本、市场需求、盈利预测等方面展开，以评估项目是否具有足够的商业前景和回报潜力。技术可行性则是考察项目技术水平和实施难度，以及技术方案是否具有可行性和可靠性。社会可行性是考虑项目对当地社会和环境的影响，以保证项目的社会效益和可持续发展。安全可行性主要考虑项目的安全性和稳定性，以确保人员财产安全和项目的可靠性。操作可行性则是针对项目实施过程中的操作和管理，以保证项目能够顺利运行并取得良好的效果。

总之，可行性研究报告是建设或科研项目前必不可少的工作，其结果将直接影响到投资决策，具有重要的实践意义。

### 3.1.1 经济可行性

首先在本系统开发之前，应该对本系统的经济可行性进行分析，对系统的成本和收益经行预估，要保证对系统开发阶段的成本和收入都有记录，虽然该系统属于独立自主设计并研发的系统，但是它也有实际的应用价值，所以可以用真实的项目去对本系统进行分析，全面深入地进行市场分析、预测。调查和预测拟建项目产品在国内、国际市场的供需情况和销售价格；研究产品的目标市场，分析市场占有率；研究确定市场，主要是产品竞争对手和自身竞争力的优势、劣势，以及产品的营销策略，并研究确定主要市场风险和风险程度，对本系统的成本进行分析，主要包括系统硬件设备的购买，系统的运行和广告的投入，和一些开发人员的成本，对一些视频版权的购买并且本系统需要在后期进行系统的维护，系统的一些bug测试等都是比较耗费成本的。

该系统的收益可以包括向用户收取一些会员费用，广告费用，一些电影的观影费，和一些电视剧的超前点播费用，可以邀请一些主播进行直播，来收取一些观众的打赏礼物的金额，可以给观众设置一些等级，例如你消费多少金额可以达到多少等级，并且会有一些专属的礼物可以赠送，并且添加一些华丽的入场特效等[12]。

### 3.1.2 技术可行性

技术可行性也称为技术风险分析，研究的对象是信息系统需要实现的功能和性能，以及技术能力约束。

本系统使用的C++语言进行开发，该语言的执行效率要快于其他语言，并且该语言的用途广泛，是面向对象语言，许多项目的底层代码都是用C++进行编写，非常适合和FFmpeg一起使用，因为FFmpeg底层库的代码就是C语言编写，C++和FFmpeg具有很好的使用配合，本系统对音视频文件的播放，使用FFmpeg库会很大的程度去提高本系统对音视频文件的解封装和解码的速度，一些主流的视频播放器也会使用FFmpeg作为底层，FFmpeg中的解封装器和解码器对于音视频文件的解封装和解码相对来说比较成熟，所以本系统用FFmpeg作为底层库会大大提高系统效率[13]。

### 3.1.3社会可行性

社会可行性涉及法律、道德、社会影响等社会因素，需要从以上方面入手进行可行性研究。

从法律上来分析，本系统是独立开发的，使用的软件功能均是面向技术开发人员免费开放的，系统所采用的技术也是业界流行的公开技术，因此在开发工具和技术方面合法合规，不会涉及到侵权行为；另外，系统开发完成后，需要选取一些网络视频、音频文件进行功能的测试，在资源的选择上，进行了精心的筛选，遵守版权规定，对于付费产品会进行购买，因此不会出现盗版等行为。

从道德方面来分析，本系统的主旨是为用户提供音视频播放工具、为用户带来开心快乐的体验，使得用户身心愉悦，同时也会借助软件申明对播放源以及资源的合法性、合规性要求，引导并要求用户播放积极乐观、健康向上的音视频资源，为其提供良好的多媒体播放体验，对用户产生正面的影响。

从社会影响方面来分析，本系统注重积极的社会影响，主要是在控制多媒体资源的质量方面以及对使用者的行为引导和规范方面，系统会进行积极的引导、提出具体的要求，通过本软件的使用，传播积极乐观的精神，传播正能量、引导人们努力工作、快乐生活，积极面对各种问题，做对社会有益的事情。

### 3.1.3安全可行性

在软件开发过程中，安全可行性是非常重要的一环。为了保护用户的人身安全和财产安全，在开发过程中需要对开发人员和用户的安全进行可行性分析。具体而言，需要对人身安全、财产安全和信息安全等方面进行分析，就本系统而言，用户使用的是电脑进行音视频播放，不会出现威胁人身安全的场景。此外，本系统是本地播放器，不会涉及到付款等财产安全问题。同时，本系统也不会要求用户提供个人信息，从而保证了信息安全。

因此，本系统在人身安全、财产安全和信息安全方面都有保证。综上所述，该系统的开发在经济、技术、社会和安全四个方面均是可行的。

## 3.2系统的需求分析

### 3.2.1 业务需求

业务需求描述了为什么要开发一个系统，通常由投资人、客户、管理者、市场或策划部门提出。使用前景和范围文档记录需求，描述组织目标、系统功能、用户、场景和环境。业务需求是开发系统的基础，需要充分分析和记录，使用文档可以提供指导。

1、业务背景

现在许多主办方选择了以远程视频的方式举办活动；大量企业采取远程办公的模式代替线下办公，而远程视频会议是远程办公不可缺少的重要组成部分；许多国家实施了严格的隔离措施，学校预计将在长时间处于封闭状态，从而导致在线教育的需求上涨……这些变化皆对音视频市场的增长产生了里程碑式的影响。2019年，全球网络实时通信市场规模为23亿美元，预计从2020年到2027年将以43.4%的复合年增长率（CAGR）增长[14]。

从2020年到2027年，印度和中国等发展中国家的互联网用户数量将不断增加，亚太地区将成为增长最快的区域市场。其中，在数字娱乐领域，中国网络视频市场、数字音乐产业市场和网络直播行业市场规模均呈现出快速增长的态势，而移动音乐行业用户规模则相对平稳。此外，实时音视频技术也在不断地应用于各种领域，例如在线教育、社交娱乐、互动电商、金融、政企服务、loT、医疗等，为人们带来更便捷的生活和更人性化的服务。

在金融银行保险行业，实时音视频技术也得到广泛应用。通过实时在线沟通，保险工作人员可以更快速地进行审核和核保，提高工作效率。此外，在其他行业和场景中，实时音视频技术也被应用于远程协作、远程医疗、虚拟演出、远程教育等方面，为人们带来更多的便利和创新。

未来，随着5G技术的广泛推广，实时音视频技术有望在工业互联网、物联网等领域成为新的增长点，为各行各业带来更多的机遇和挑战。音视频发展行情图。

2、业务需求：

从用户角度分析：用户可以打开本系统，通过点击想要播放的本地音视频文件可以在本系统进行播放，用户可以根据自身的意愿去选择音量的大小，并且可按静音按钮选择静音，并且用户可以去跳动进度条去选择想要播放的位置，可以按暂停按钮去让视频的播放停止，按播放按钮让视频继续播放，可以按停止按钮来去重新选择播放的文件。

### 3.2.2 系统需求

系统需求主要分为功能性需求和非功能性需求。

1、功能性需求是指规定开发人员必须在产品中实现的软件功能，以便用户能够利用这些功能来完成任务，满足业务需求。它描述了系统应该具备的功能，因此通常用“应该”来进行描述，例如“系统应该发送电子邮件来通知用户已接受其预定”，功能需求是软件开发过程中非常重要的一部分，它描述了系统需要实现的功能和行为，是开发人员和用户沟通的桥梁，也是软件产品开发的基础。

本系统可以分为本地播放和在线播放，本地播放可以打开目标文件，如音频、视频文件。通过音视频的解码和渲染播放音频、视频文件。音视频解码技术主要包括： H.264/AVC是一种广泛使用的视频编码标准，也是目前最先进的视频解码技术之一。它具有出色的压缩比和图像质量，在高清视频传输、视频会议等领域得到广泛应用。HEVC/H.265是一种新一代视频编码标准，与H.264相比，它可以实现更高的压缩比和更好的图像质量，适合于4K、8K超高清视频传输等场景。AAC是一种高级音频编码标准，可以提供更高的音频质量和更低的比特率，适用于移动设备、数字音乐播放器等场景。MP3是一种流行的音频编码格式，具有较高的压缩比和广泛的兼容性，适合于网络传输和储存。AC-3和DTS是两种常用的多声道音频编码标准，用于影视制作和家庭影院等领域。可以通过暂停按钮去暂停正在播放的音视频文件，也可以直接终止音视频文件的播放，重新播放新的音视频文件,在感觉音量较大的时候可以调节音量的大小,如果想要调整音视频的播放位置也可以直接使用进度条去调整。

在线播放主要是对一些流媒体文件推流流媒体服务器中在通过拉流到客户端的播放器中，其中主要流程为：

(1)视频采集：使用视频采集设备，如摄像头或屏幕录制软件等，将视觉内容采集下来。同时，采集音频信号，例如使用话筒或麦克风，以获取声音。

(2)编码：将采集到的视频和音频数据进行压缩编码，主要包括视频编码和音频编码。常见的视频编码格式有H.264、HEVC/H.265等，音频编码格式有AAC、MP3等。

(3)传输：将编码后的视频和音频数据通过网络传输到服务器端。可以使用HTTP、RTMP、WebRTC等协议进行传输。

(4)服务端处理：在服务器端进行解码和处理，将接收到的音视频数据进行解码，并对其进行转换和存储，以便于可视化播放。

(5)客户端播放：在客户端上播放音视频流，通常使用专业的音视频播放器或浏览器内置的播放器，在用户的设备上实时解码和显示。

2、非功能性需求：系统需求中还包括一些潜在的要求具备的需求。

(1)统应该确保播放视频的流畅性，避免出现卡顿现象，让用户等待，降低用户体验。

(2)系统属于多线程，应该主要线程安全问题。

(3)系统应该在多个操作系统下能够运行。

(4)系统应该确定每个按键的正常使用避免出现按键不好使的情况。

(5)确保在播放视频seek时不会出现，视频画面不更新或者直接系统崩溃的可能。

(6)系统应该确保他的可维护性。

(7)系统属于本地播放器，在没有网络的情况下也可以使用。

系统用例图如3-1所示。

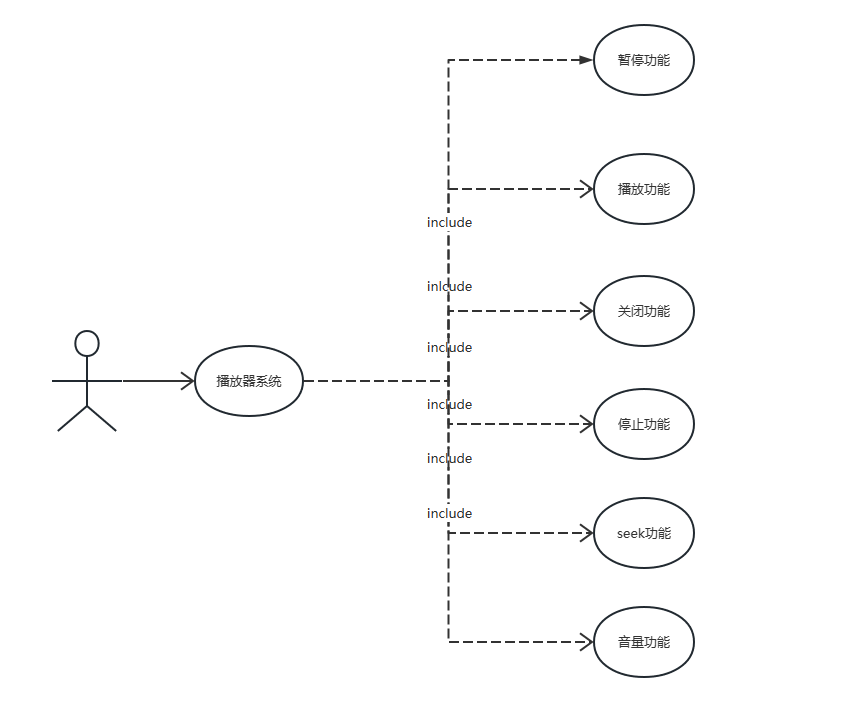


图3-1系统用例图

# 第4章 音视频播放系统总体设计

## 4.1音视频播放系统架构总体设计

本系统采用Qt作为窗体，实现了播放系统前端的展示，通过调用FFmpeg库函数实现了对本地文件播放，首先对播放器传入封装好的文件，播放器通过将封装好的文件传入FFmpeg解复用器中实现对流媒体文件的解复用，先按照给出的文件流类型，选择容器解析和demux解析。若无文件流类型，则使用探测的方式尝试。解析容器，使用软件ffmpeg提供的或其他插件提供等，软件解码依赖ffmpeg提供库或其他开放的源代码解码器，在嵌入式设备中，则存在使用硬件demux进行音视频分离的场景，比如机顶盒TS流的demux，之所以需要demux，是因为音视频在制作的时候实际上都是独立编码的，得到的是分开的数据，为了传输方便必须要用某种方式合起来，这就有了各种封装格式，也就有了demux。

再将解复用后的音频流和视频流，通过FFmpeg库函数读取出在音频流和视频中的数据包，再将数据包分别送入对应的音频解码线程和视频解码线程，根据判断硬件支持情况，选择使用软硬件解码器，视频都使用硬件解码器，有些独立的音频播放可以使用软件解码器(mpeg audio layer 1/2/3 (mpga))。对于加扰的视频需另外增加处理，软件解码依赖FFmpeg提供库或其他开放的源代码解码器，如果使用硬件decoder, 则需要注意其硬件接口。

再将解码后的音频数据PCM，进行重采样将PCM的数据格式变成和SDL要求的数据格式相同的数据，将重采样后的PCM数据送进SDL音频音频缓冲区通过SDL音频缓冲区将音频数据播放出来，将解码出来的视频数据YUV，转换成RGB格式后，通过Qt幕布实现了对视频数据的绘制播放，最后通过音视频同步策略实现对音视频同步[15]。

音视频同步是指在播放音视频时，确保画面和声音的同步，这是提供良好用户体验的基本要求。为了实现音视频同步，可以采用三种方式：

1、以视频为基准，同步音频到视频：这种方式主要是通过调整音频的播放速度来实现同步。如果视频播放速度慢了，可以加快音频的播放速度或者丢掉部分音频帧；如果音频播放速度快了，则可以放慢音频的播放速度。这种方式适用于视频播放速度比较稳定的情况。

2、以音频为基准，同步视频到音频：这种方式主要是通过调整视频帧的播放速度来实现同步。如果视频播放速度慢了，可以加快视频帧的播放速度或者丢掉部分视频帧；如果视频播放速度快了，则可以延迟播放视频帧，继续渲染上一帧。这种方式适用于视频播放速度比较不稳定的情况。

3、以外部时钟为基准，同步音视频到外部时钟：这种方式主要是通过外部时钟来同步音视频。根据外部时钟的变化，调整音视频的播放速度，以确保音视频同步。这种方式主要适用于TS流等需要依赖外部时钟同步的情况。本系统的架构设计图如图4-1所示。

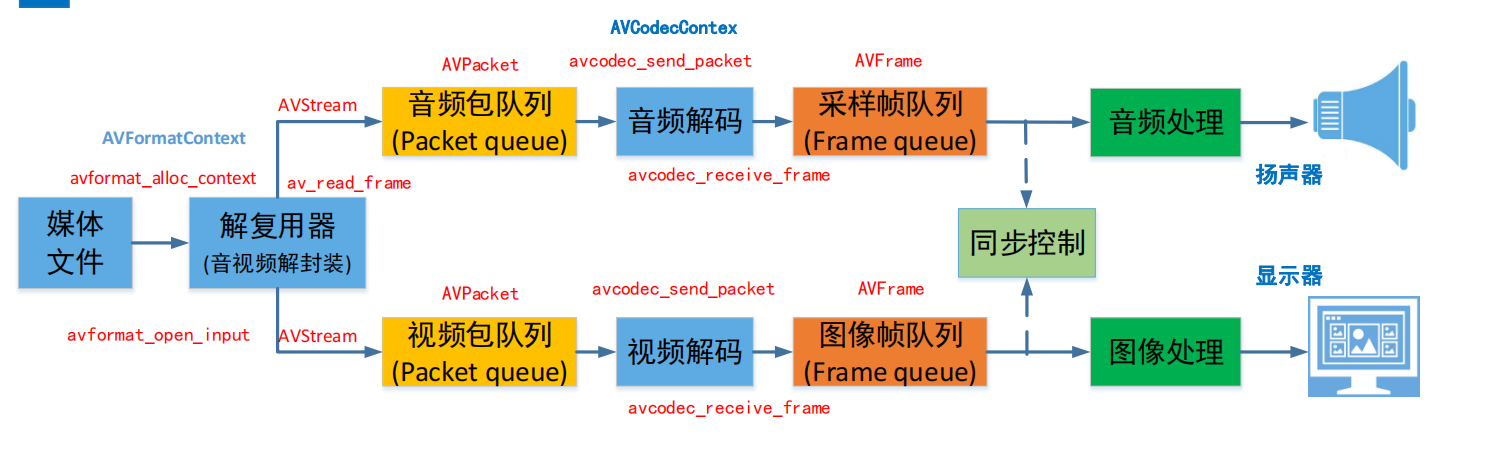


图4-1 本系统架构图

## 4.2 音视频播放系统功能的总体设计

本系统设计为7个模块分别为打开模块，关闭模块，播放模块，暂停模块，停止模块，音量模块和seek模块，以下开始对各个模块进行描述。

1、打开模块：打开目标文件，如音频、视频文件，通过点击QT中的打开文件按钮实现对本地音视频的播放

2、关闭模块：关闭播放器。

3、播放模块:播放音频、视频文件，打开文件后见文件进行解复用，解码的一些列工作实现对文件的播放功能。

4、暂停模块：暂时停止正在播放的音/视频文件，通过点击暂停按钮将状态设置位Pause实现了音频和视频解码的暂停。

5、停止模块：结束当前音/视频文件的播放，但不退出播放器，将播放状态设置位Stopping，释放音频和视频队列中的内存实现对音视频文件播放的停止。

6、音量模块：调节音量大小。

7、seek模块：调整音视频播放位置，通过调用FFmpeg库函数中的av\_seek\_frame实现对音视频的seek操作。

该系统的功能模块图如4-2所示。

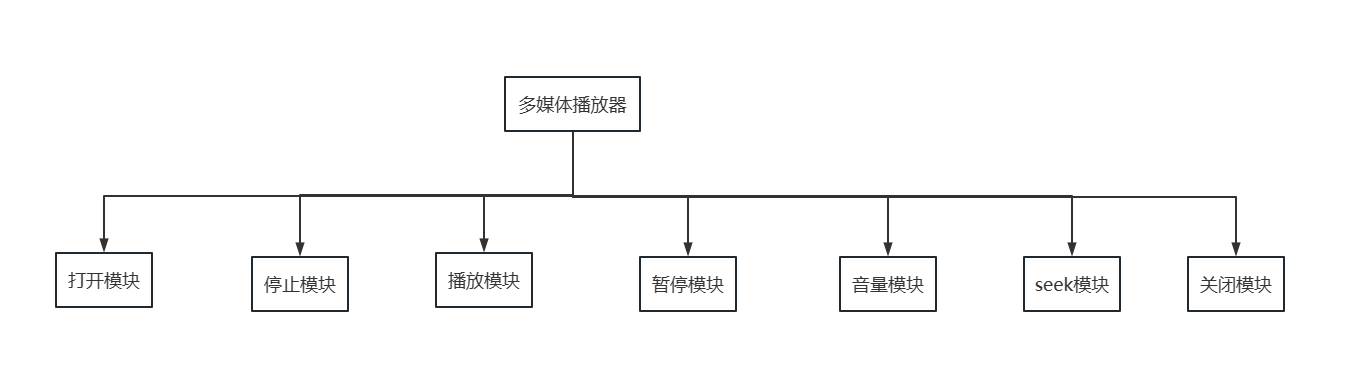


图4-2 功能模块图

## 4.3 音视频文件到播放总体设计

### 4.3.1 音视频文件解封装

音视频文件解封装示意图，如图4-3所示

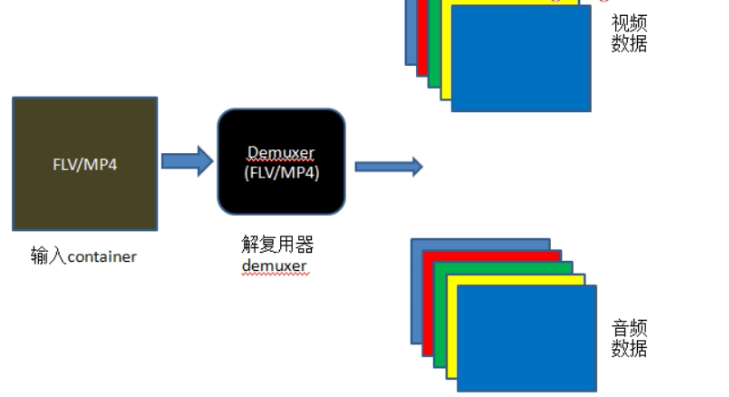


图4-3解封装图

从图中可以看出大致的解封装流程：首先要对解复用器进行初始化，其次,将输入的封装格式文件给到解复用器内，最后利用解封装对 Container 进行解封装。首先对输入文件(Container 文件)、输出文件(Video/Audio 进行理),方便后面的使用；其次打开输入文件，并分配 Format Context，从输入文件中得到流信息，之后打开视频、音频编码器 Context,针对视频数据，分配图像。分配 frame 结构，初始化 packet，从输入文件中读取 frame 信息，并之后进行解码 packet。最后释放各种分配的数据信息。在用FFmpeg的API对音视频文件进行解封装时应该了解一些他的相关格式：

1、容器格式：不管是音频文件还是视频格式的文件，都是一个多媒体的容

器，即container，比如常见的视频容器格式有avi、mp4、mkv、flv、rm/rmvb、mov、ts、vob、dat，音频容器格式有MP3、WAV、AAC、APE，FLAC等等，它容纳了视频、音频、字幕（subtitle）等一个或多个基本流数据，有的甚至一个容器中存放有多个视频、音频以及字幕。

2、音视频压缩格式是对基本流进行压缩的方式，常见的视频压缩格式包括 MPEG-2、MPEG-4、H.264、VC-1、RM，常见的音频压缩格式包括 MPA、AAC、AC3、DTS。压缩可以减小存储空间，比如 MPEG-2 的压缩比可以达到 25:1，而 H.264 可以达到 102:1。

3、ES，即 Elementary Stream，是指单独的一路视频、一条音频、一个字幕或者单个的附加数据。常见的多媒体文件包括一个视频 ES、一个音频 ES，有的还包含多个视频 ES 和音频 ES 以及字幕 ES。蓝光原版的 TS 一般包含多个音轨 ES 和字幕 ES，但不是所有字幕都有字幕 ES，有些字幕已经内嵌进视频。

4、Demux 是指将视音频以及字幕等基本流分离出来的过程，也称为解封装或解复用。分离出来的各个基本流可以分别送给视频解码器、音频解码器等解码后才能得到图像声音。

5、Remux 是指将基本的音频、视频、字幕等组合成一个完整的多媒体的过程，也称为封装或复用。比如电影网站的音视频压制人员需要先进行 Demux，分离成 ES，然后加入必要的中文字幕和音轨后，重新封装。所有的转码工具也都必须有 Remux 和重新 Demux 的过程。复用与解复用的概念对于熟悉 DVB 行业的读者来说应该比较清楚。

6、PTS，即显示时间戳，指图像或者声音在解码后应该显示或者发声的时间点。音视频不是一解码出来就播出来，需要靠 PTS 来同步，以确保视频和音频的同步播放。音视频文件解封装流程，如图4-4所示。

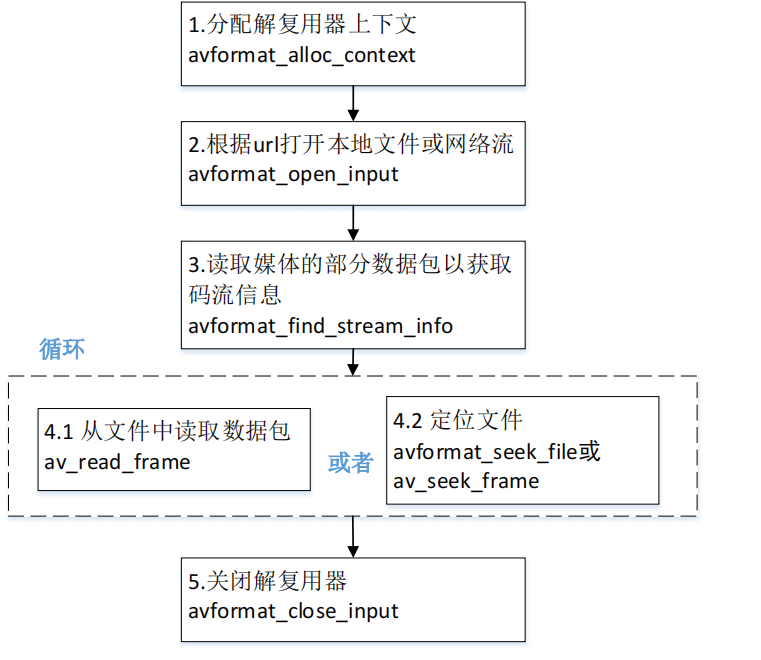


图4-4音视频解封装流程图

### 音频解码

本系统的音频解码主要是接收来自解复用器将音视频文件解复用的音频流中的数据包，将音频流中的数据包发送到FFmpeg解码器中，实现对音频数据的解码，主要采用了一些FFmped的API。其主要API的调用如下。

1、avcodec\_find\_decoder：根据指定的AVCodecID查找注册的解码器。

2、av\_parser\_init：初始化AVCodecParserContext。

3、avcodec\_alloc\_context3：为AVCodecContext分配内存。

4、avcodec\_open2：打开解码器。

5、av\_parser\_parse2：解析获得⼀个Packet。

6、avcodec\_send\_packet：将AVPacket压缩数据给解码器。

7、avcodec\_receive\_frame：获取到解码后的AVFrame数据。

8、av\_get\_bytes\_per\_sample: 获取每个sample中的字节数。

libavcodec可能不得不对输入的数据进行拷贝。在一个循环体内去接收到codec的输出，即周期性地调用avcodec receive来接收codec 输出的数据，解码，调用avcode\_receive\_ frame，如果成功会返回一个包含未压缩数据Frame。反复地调用avcodec\_ receive\_ packet直到返回AVERROR(EAGAIN)或其他错误。返回AVERROR(EAGAIN)错误表示codec需要新的输入来输出更多的数据。对于每个输入的packet 或frame，codec一般会输出一个frame或packet，但是也有可能输出0个或者多于1个。流处理结束的时候需要flush (冲刷)解码器codec。因为codec 可能在内部缓冲多个frame或packet,出于性能或其他必要的情况(如考虑B帧的情况)。处理流程如下:调用avcodec\_send \*0传入的Frame或Packet指针设置为NULL。这将进 入draining mode (排水模式)。反复地调用avcodec\_receive\_ \*()直 到返回AVERROR EOF,该方法在.draining mode时不会返回AVERROR(EAGAIN)的错误，除非你没有进入draining mode。当重新开启codec时，需要先调用avcodec\_flush\_buffer来重置codec。

音频解码流程图，如图4-5所示。

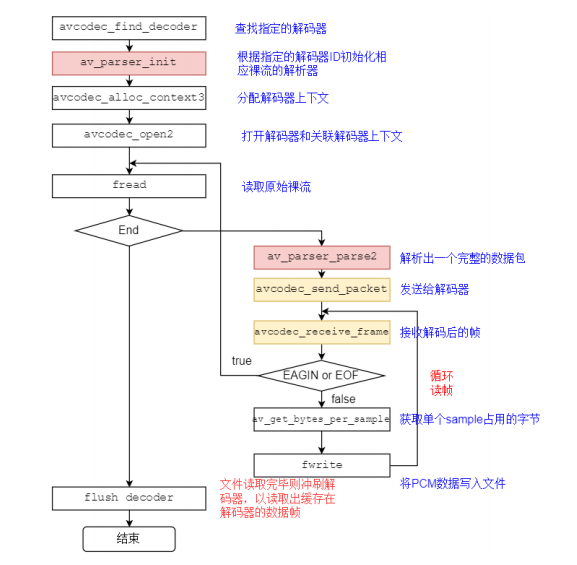


图4-5音频解码流程图

### 视频解码

编解码器（CODEC）：能够进行视频和音频压缩（CO）与解压缩（DEC），是视频编解码的核心部分。容器/多媒体文件（Container/File）：没有了解视频的编解码之前，总是错误的认为平常下载的电影的文件的后缀（avi，mkv，rmvb等）就是视频的编码方式。事实上，刚才提到的几种文件的后缀，并不是视频的编码方式，只是其封装的方式。一个视频文件通常有视频数据、音频数据以及字幕等，封装的格式决定这些数据在文件中是如何的存放的，封装在一起音频、视频等数据组成的多媒体文件，也可以叫做容器（其中包含了视音频数据）。所以，只看多媒体文件的后缀名是难以知道视音频的编码方式的。

FFmpeg的视频解码过程可以简要概括为以下几个步骤：注册所支持的文件格式及其对应的编解码器。打开待解码的视频文件或网络流。从文件头部读取流相关的信息。找到视频流。查找视频流对应的解码器。打开视频解码器。为解码帧分配内存。从视频流中读取数据到Packet中。对视频帧进行解码。解码过程的具体如下：

1、注册

av\_register\_all该函数注册支持的所有的文件格式（容器）及其对应的CODEC，只需要调用一次，故一般放在main函数中。也可以注册某个特定的容器格式，但通常来说不需要这么做。

2、打开文件

avformat\_open\_input该函数读取文件的头信息，并将其信息保存到AVFormatContext结构体中。

3、获取必要的CODEC参数

avformat\_open\_input通过解析多媒体文件或流的头信息及其他的辅助数据，能够获取到足够多的关于文件、流和CODEC的信息，并将这些信息填充到AVFormatContext结构体中。但任何一种多媒体格式（容器）提供的信息都是有限的，而且不同的多媒体制作软件对头信息的设置也不尽相同，在制作多媒体文件的时候难免会引入一些错误。也就是说，仅仅通过avformat\_open\_input并不能保证能够获取所需要的信息，所以一般要使用以下代码实现：

avformat\_find\_stream\_info(AVFormatContext \*ic, AVDictionary \*\*options)

avformat\_find\_stream\_info主要用来获取必要的CODEC参数，设置到ic->streams[i]->codec。

4、打开解码器

经过上面的步骤，已经将文件格式信息读取到了AVFormatContext中，要打开流数据相应的CODEC，找到视频流 video stream一个多媒体文件包含有多个原始流。

5、读取数据帧并解码

已经有了相应的解码器，下面的工作就是将数据从流中读出，并解码为没有压缩的原始数据。

视频解码流程图如图4-6所示

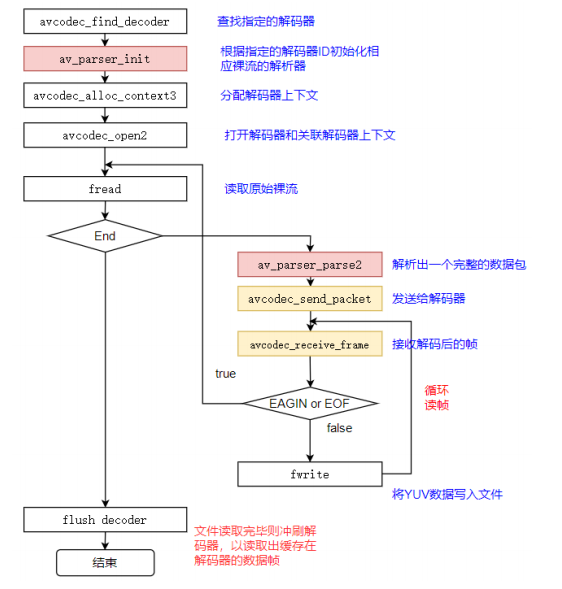


图4-6视频解码流程图

### 音频重采样

本系统解码后出来的音频数据PCM的格式将会和SDL要求的音频数据的格式不符合，这样SDL就没有办法对音频进行播放，所以在进行音频播放之前要对音频数据进行一次重采样。

在进行音视频开发中，遇到的很多设备的对于音频的格式要求是不一样的，例如在一个项目中遇到过这么一个情形：正常的电脑播放声音的音频是2声道16位44.1khz或者48khz的PCM音频格式才能够播放，但是此时项目中需要使用的一款usb耳机它的格式如下图所示:单声道16位44.1khz或者48khz。通常情况下在双声道16位的音频设备需要输入的肯定是对应双声道16位的音频数据才能够正确播放，这个时候来了个单声道16位的数据，播放时声音肯定不是想要的了。这个时候想要解决这个问题就是需要对这个麦克风的音频数据进行重采样由原来的单声道进行重采样为双声道的数据，以此能够在正常的扬声器中播放音频。

一般情况下重采样能够满足如下需求：

1. 采样频率转换:例如44.1khz变为48khz等。
2. 声道数变换：例如单双声道的数据转换等。
3. 音频格式变换:例如16位转8位等。

重采样流程：在使用ffmpeg或者其他的重采样方式时，无外乎都是需要注意上面所介绍过的采样频率、声道数以及音频格式的变化等，同时还有一个特别需要注意的是送入样本点和输出样本点的所占的字节数以及保存方式，这些都是需要考虑的，因此每次进行重采样时对输入和输出的样本字节数的计算是特别需要注意的，否则会导致重采样得到的结果不是所期望的。

### 音视频同步

在本系统中因为音视频是通过将数据传递到SDL缓冲区中再进行播放，这样音频的输出频率就是确定的，所以本系统采用的是以音频为基准的策略对音视频进行同步。

在视频播放过程中，可以使用上一帧的PTS和设置的播放上一帧时的延迟时间来估算出当前帧的延迟时间。通过比较当前帧的PTS和音频时钟，可以判断视频播放速度是否快了或慢了。根据判断结果，可以调整播放下一帧的延迟时间，以实现视音频的同步播放。

具体而言，可以用frame\_last\_pts和frame\_last\_delay来保存上一帧的PTS以及设置的播放上一帧时的延迟时间。在播放当前帧时，根据当前帧的PTS和上一帧的PTS差来估算播放当前帧的延迟时间。然后，将当前帧的PTS和音频时钟相比较，判断视频播放速度是否快了还是慢了。如果视频播放过快，则加倍延迟时间；如果视频播放过慢，则将延迟时间设置为0。frame\_timer保存着视频播放的延迟时间总和，将其与当前时间点的差值作为播放下一帧的真正延迟时间。

# 第5章 音视频播放系统的详细设计与实现

## 5.1音视频主线程模块的详细设计与实现

### 5.1.1 音视频文件解封装功能

音视频文件解封装，也称为音视频解复用，是指将包含音频和视频流的容器格式文件（如MP4、AVI、FLV等）中的音频和视频数据分离出来，并提取相关元数据信息的过程。音视频文件解封装可以获取到音频和视频流，从而进行后续的编解码、处理、编辑等操作。

具体而言，实现音视频文件解封装需要以下步骤：

1、打开文件：打开需要解封装的音视频文件。

2、读取文件头：读取容器格式文件的文件头，并根据文件头信息确定音频和视频流所在位置。

3、解析元数据：解析容器中的元数据信息，例如视频分辨率、帧率、音频采样率、声道数等信息。

4、分离音视频流：根据文件头信息和元数据信息，分离音频和视频流，并将它们保存为单独的文件或内存中的数据流。

本系统通过调用QFileDialog中的getOpenFileName函数去获取要打开的文件名称，自定义函数setFilename(QString &filename)将获取到的文件名称传入函数，将获取到的文件名称复制到自定义的字符串数组\_filename中，再将本系统的\_state改成Playing状态，并调用readFile()函数启动解封装功能，通过avformat\_open\_input方法创建解封装上下文，音频流核视频流数据保存到解封装上下文中，再调用检索函数avformat\_find\_stream\_info(\_fmtCtx, nullptr)检索上下文中的流信息，检索完流信息后将音频信息和视频信息进行初始化；将播放状态进行更改，通过调用setState(Playing)和av\_read\_frame(\_fmtCtx,&pkt)将检索出来的流信息保存到Packet的封装数据包中，判断数据报的索引来判断是音频数据包还是视频数据报，如果是音频数据包加到音频缓冲区，如果是视频数据包加到视频缓冲区。

### 5.1.2 调节音量功能

本系统可以通过调节进度条控制音量大小，通过进度条的x值来计算出音量value，计算方法如下：value=minimum()+(ev->pos().x()\*1.0/width())\*(maximum()-minimum())

通过x和width的值计算出相对的比例，再通过相对的比例和ValueRang的值计算获取value值。调节进度条的实现是通过setValue(value)设置对应的值。首先获取到进度条对应的值，计算出对应的比例和SDL中的最大音量值的成绩，从而获取到当前的音量值，最后设置SDL音量，并将音量设计好后在数据包发送到SDL音频缓存队列。

通过点击静音按钮，可以将\_mute设置成true去将音量设置成0，实现音量的静音功能。音量功能图如图5-1所示。



图5-1音量功能图

### 5.1.3 获取视频总时长功能

通过转换函数可以将FFmpeg的时间戳改成想要的现实时间，并在Lable中显示，核心代码如下：

getDuration(){\_fmtCtx ? round(\_fmtCtx->duration \* av\_q2d(AV\_TIME\_BASE\_Q)): 0;}，

解封装初始化完成后，将发送一个信号emit initFinished(this)，mainwindow在接收这个信号后采用对应的onPlayerInitFinshed槽函数去处理信号，通过槽函数调用getDuration获取到进度条的范围和对应的总时长，并显示到Lable中，最后通过getTimeText将获取到的时长按照hh-mm-ss显示。核心代码如下：

ui->timeSlider->setRange(0,duration);

ui->durationLable->setText (getTimeText(dura- tion ));

QLatin1Char fill=QLatin1-Char('0');

Return QString("%1:%2:%3").arg(seconds/3600,2,10,fill).arg((seconds/60)%60, 2, 10, fill). arg(seconds%60,2,10,fill);

视频总时长显示图，如图5-2所示。



图5-2视频总时长显示图

### 5.1.4获取当前时长功能

因为本系统是以音频为基准做音视频同步，所以本系统将获取到音频的时间去做Slider的改变以及Label显示的改变。在播放时间改变时音频线程将对外发送时间改变的信号emit timeChanged(this)，mainwindow中的槽函数onPlayer-TimeChange()接收到信号后，将对进度条的值做更改，调用getTime函数获取到当前音频播放时间，改变进度条的值；同时，系统的label也应该做相应的改变。当前播放时间图如图5-3所示。



图5-3当前播放时间图

### 5.1.5改变视频播放状态功能

本系统存在3种状态，Playing播放状态，Pause暂停状态，Stopped停止状态，在打开文件的时候应该将本系统的状态改成播放状态Playing，在点击暂停后本系统的状态应改为暂停状态Pause，在点击停止后本系统的状态应该为停止状态Stopped，可以通过setState(State state)去设置状态，比较现在的状态和要设置的状态是否为同一状态，为同一状态直接返回，如果状态不为同一状态要求发送状态改变信号emit stateChanged(this)，通过槽函数onPlayerStateChange实现相应的状态转换。

设置系统播放功能时，可以通过getState方法获取到本系统现在的状态，并通过play()函数控制系统的播放功能，核心代码如下：

if (\_state == Playing) return;

if (\_state == Stopped) {

std::thread([this](){readFile();}).detach();

}else{setState(Playing);}

通过stop()设置系统的停止功能代码如下：

if (\_state == Stopped) return;

\_state = Stopped;

emit stateChanged(this);

如果系统的状态是Stopped直接返回如果系统的状态是其他状态应该系统的状态改成Stopped再去发送改变状态的信号。通过pause()设置系统暂停功能，代码如下：

if (\_state != Playing) return;

setState(Paused);

### 5.1.6初始化解码器功能

本系统完成对封装文件的解码后，将找到对应的音频流和视频流，并将音频流和视频流放入对应的Packet的压缩数据包，本系统通过对应的解码器将数据包中的相对应的视频数据和音频数据从压缩数据包中提取出来，在主线程中应该对解码器进行初始化，再通过对应的音频线程和视频线程找到对应的音频解码器和视频解码器。

通过av\_find\_best\_stream找到最适合的流信息，将找的的流索引返回到ret中，通过在流中找到的索引streamIndex，借助流索引\*stream = \_fmtCtx->streams[streamIdx]找到对应的流，借助对应的流找到解码器的id，通过解码器id在FFmpeg中去寻找对应的解码器，再创建解码器上下文，为解码器上下文分配空间，将流中解码器的参数拷贝到解码器上下文中，最后去打开解码器去对解码器进行应用，代码如下：

decoder = avcodec\_find\_decoder ((\*stream)->codecpar->codec\_id)\*decodeCtx = avcodec\_alloc\_context3(decoder)

avcodec\_parameters\_to\_context(\*decodeCtx, (\*stream) -> codecpar)

avcodec\_open2(\*decodeCtx, decoder, nullptr)

## 5.2音频线程模块的详细设计与实现

### 5.2.1 解码音频功能

音频解码是将压缩后的音频文件（如MP3、AAC等格式）还原成无损音频数据的过程。压缩音频文件可以减小文件大小，提高传输效率，但同时也会导致音质降低。因此，在进行音频播放或编辑时需要对压缩文件进行解码，将其中的音频数据还原成PCM（脉冲编码调制）格式的音频流。

本系统通过将解封装后的音频数据放入到音频缓冲区中，再通过音频解码线程将缓冲区中的音频数据读取后进行解码，采用avcodec\_send\_packet(\_aDecodeCtx, &pkt)方法将音频数据包发送到解码器进行解码。具体的解码过程如下：

步骤1：由于压缩后的音频数据是由多个帧（frame）组成的，每个帧包含有若干个子帧（subframe），每个子帧包含有一个或多个独立的音频通道。因此解码器需要先从文件头中读取采样率、bitrate等参数；

步骤2：根据参数对数据进行解码，并进行反量化。对每个子帧进行反变换（如MDCT）和重叠添加，得到频域数据。

步骤3：应用掩蔽模型（Masking Model）将频域数据转换为时域数据，并进行重新量化。

步骤4：将多个子帧合并成PCM音频数据流，以供后续处理或输出。

步骤5：将解码后的数据调用avcodec\_receive\_frame (\_aDecodeCtx, \_aSwrInFrame)函数，把保存在\_aDecodeCtx中的解码后的音频信息输出到\_aSwrInFrame中再通过音频的重采样功能，将音频的格式设置为和SDL缓冲区播放的音频相同，最后通过将音频数据发送到SDL缓冲在进行音频的播放功能。

步骤6：播放完成后系统将调用freeAudio()清空缓冲区的资源防止内存泄漏。音频解码过程图，如图5-4所示。

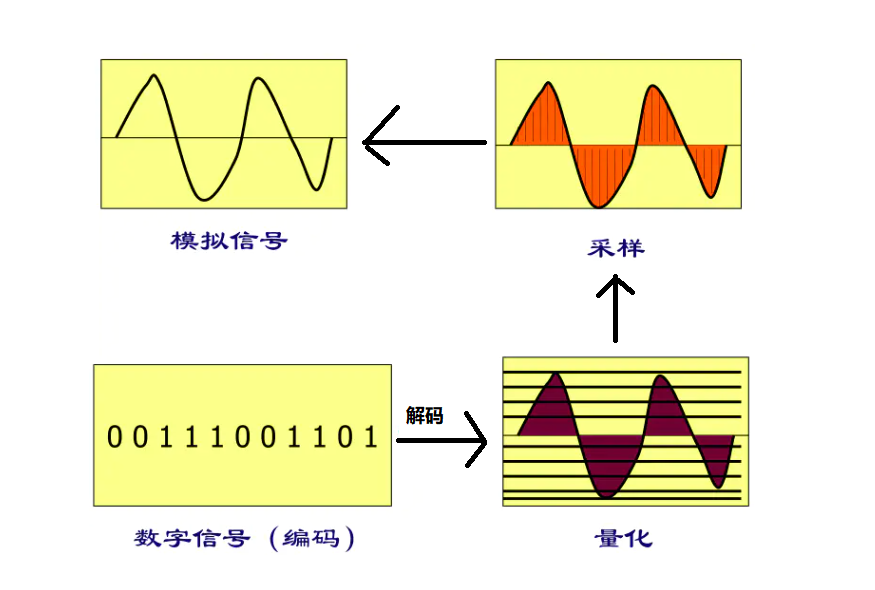


图5-4音频解码过程图

### 5.2.2音频重采样功能

重采样是指将一个数字信号从一个采样率转换为另一个采样率的过程。在音频处理中，重采样通常用于将不同采样率的音频文件进行格式转换、匹配设备、增加/降低音频质量等应用，因为本系统在解码出的音视频数据格式为Float类型与SDL缓冲区播放的音频数据格式S16不匹配，所以应该将本系统解码出来的音频数据进行重采样成和SDL缓冲区播放的音频数据类型相一致。

重采样的具体操有插值和抽取两种其中，插值为在从高采样率到低采样率的情况下，需要插入新的采样点以保留更多的信息。插值方法包括零阶插值（即将相邻采样点之间的距离等分），线性插值（即使用两个相邻采样点之间的斜率来计算中间位置的采样值），三次样条插值（即使用一条三次多项式逼近相邻采样点之间的曲线）等。抽取为在从低采样率到高采样率的情况下，需要抽取原始信号中的采样点以匹配目标采样率。抽取方法包括最近邻抽取（即选择最近的一个采样点作为新的采样值），线性插值（即使用两个相邻采样点之间的斜率来计算中间位置的采样值）等。

定义音频重采样的输出参数实现重采样功能，具体参数如下：

1. 重采样的音频输出格式：aSwrOutSpec.sampleFmt= AV\_SAMPLE \_FMT\_S16
2. 音频重采样输出的采样率：\_aSwrOutSpec.sampleRate=44100
3. 音频输出声道布局：\_aSwrOutSpec.chLayout= AV\_CH\_LAYOUT\_ STEREO
4. 音频重采样输出的声道数：\_aSwrOutSpec.chs= av\_get\_channel\_layout\_nb\_ channels (\_aSwrOut- Spec.chLayout)
5. 样本大小：\_aSwrOutSpec. bytesPerSampe- Frame

通过swr\_alloc\_set\_opts()将输出参数保存到重采样上下文中，求出重采样真正成功转换的样本数量，再通过ret\*\_aSwrOutSpec.bytesPerSampeFrame计算出重采样样本大小。使用SDL的回调函数获取到样本大小也即解码出的PCM数据的大小，最后在分段的放入SDL缓冲区，进行音频播放。

### 5.2.3 播放音频功能

播放音频的具体流程为：首先需要打开音频文件，并从文件中读取音频数据。根据音频数据的格式、采样率、声道数等参数，配置音频设备的参数，以确保能够正确地解码和播放音频数据。使用操作系统提供的API或第三方库函数打开音频设备，并将音频参数传递给设备。在音频设备开始播放之前，需要将音频数据填充到音频缓冲区中。音频缓冲区通常由一个环形缓冲区实现，应尽可能地减少缓冲区的延迟，以确保音频播放的连续性和实时性。当音频缓冲区被填满后，音频设备会开始播放音频数据。在音频播放过程中，需要不断地从音频文件中读取数据并填充到音频缓冲区中，以确保音频播放的连续性，为了提高音频播放的质量和效果，还可以使用音频增益、混响、均衡器等音频处理算法进行优化

SDL的回调函数sdlAudioCallbackFunc()会被周期性地调用，读取指定数量的音频数据并填充到音频缓冲区中，在填充到音频缓冲时应该注意，当前音频数据的索引是否大于要填充的音频数据大小，如果大于if(\_aSwrOutIdx>=\_aSwrOutSize)，要使用\_aSwrOutSize=decodeAudio()重新获取PCM数据，并通过\_aSwrOutIdx=0操作将索引清0，当没有解码出来PCM数据时要进行静音处理，音频数据索引小于填充音频数据大小时，要对音频数据进行填充，用填充剩下的音频数据和要填充的长度作对比选取小的长度作为要填充的长度，通过SDL\_MixAudio()函数将已经准备好的PCM数据填充到音频缓冲区中，通过SDL\_OpenAudio(&spec, nullptr)去打开音频设备，再调用SDL\_PauseAudio(0)函数开始播放音频。

## 5.3视频线程模块的详细设计与实现

### 5.3.1 解码视频功能

视频解码是指将压缩后的视频数据（例如H.264、AVC等格式）转换为可渲染的图像序列的过程。对于每一帧视频数据，需要首先进行解析，获取其中的关键信息，例如帧类型（I帧、P帧、B帧等）、图像大小、采样率、色彩空间等参数。对于非关键帧，视频数据通常采用DCT（离散余弦变换）等算法进行压缩。在解码时，需要使用相应的算法进行反向变换，以恢复原始的像素数据。对于每一帧视频数据，根据其类型及相关参数，选择相应的解码器进行解码，并对数据进行降噪、增强、锐化等处理，以提高视频质量和清晰度。将解码后的视频帧传递给渲染引擎，在解码完成后，可以将得到的视频帧数据传递给渲染引擎进行渲染。渲染引擎通常使用硬件加速的图形API（例如OpenGL、DirectX等）来实现视频帧的展示。

本系统将放在视频缓冲区的数据包取出，在解码视频数据时应该判断本系统的播放状态，有以下几种情况：

1. 如果状态为Pause暂停状态，应该停止解码；
2. 如果状态为Stopped停止状态应该释放相应的资源
3. 如果为播放状态可以继续进行解码的功能，再通过将视频数据包发送到视频解码器进行解码，具体的解码代码如下：

avcodec\_send\_packet(\_vDecodeCtx, &pkt)。

状态判断好后，通过下面几种算法实现对视频数据进行解码功能：

1. 离散余弦变换（DCT）：对于视频编码中使用的帧间预测技术，通常会采用DCT算法进行压缩。在解码时，需要使用相应的反向DCT算法进行解压缩，以恢复原始的像素数据；
2. 运动估计和补偿算法：在视频编码中，为了提高压缩率和减少冗余数据，通常会采用运动估计和补偿技术。在解码时，需要使用相应的运动估计算法来确定图像中物体的运动轨迹，并使用运动补偿算法对帧间差异进行处理；
3. 零树编码（Zero-tree coding）：零树编码是一种广泛应用于视频编码中的无损压缩算法。在解码时，需要使用零树解码算法来还原原始的像素数据；
4. 反熵编码算法：视频编码中通常会采用一些熵编码算法，例如哈夫曼编码、算术编码等。在解码时，需要使用相应的反熵编码算法来还原原始的像素数据；
5. 降噪和增强算法：为了提高视频质量和清晰度，视频解码器通常会使用一些降噪、增强、锐化等算法进行处理。例如，可以采用基于小波变换的去噪算法来减少图像中的噪声，或者采用直方图均衡化算法来提高图像的对比度；

最后，通过avcodec\_receive\_frame()获取到解码后的视频数据。系统解码过后的视频数据是无法直接使用的，因为解码出来的视频数据时YUV的格式，而要在幕布上播放的视频数据应该是RGB格式，所以应该将解码出来的视频数据进行格式转换，将视频数据转换成可播放的RGB视频数据格式。

视频解码流程图，如图5-5所示。

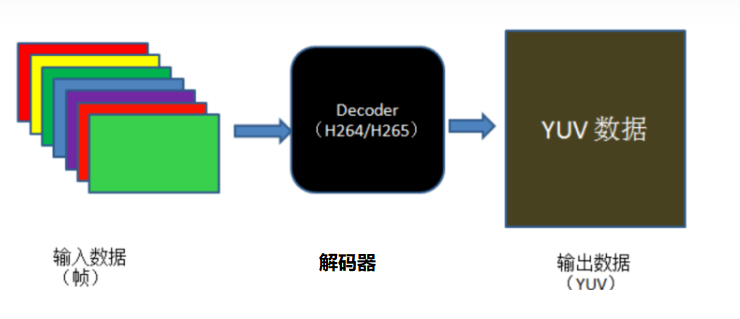


图5-5视频解码流程图

### 5.3.2改变视频像素格式功能

YUV转RGB是将一种颜色空间的视频像素格式（通常是YUV）转换为另一种（通常是RGB）。这种转换涉及到对每个像素的亮度、色彩和饱和度进行重新计算和调整。

在YUV到RGB的转换中，首先需要将YUV值转换为RGB分量。这可以通过使用矩阵乘法来完成，其中一个3x3的矩阵被应用于U、V值，而另一个常数向量被加到结果中以计算最终的RGB值。具体来说，以下是转换计算方法：

R = Y + 1.13983V

G = Y - 0.39465U - 0.58060V

B = Y + 2.03211U

其中Y、U和V是输入YUV值，而R、G和B是输出RGB值。

在进行像素格式转换时本系统要设计好每一个视频帧要输出的参数，本系统将解码出来的宽高进行处理让宽高的数值为4的倍数因为QImage是按照四个字节对齐的，当传入的图像数据的宽和高不是四的倍数的时候，它会自动补齐成四字节对齐的方式。比如，视频的宽度是1025，不是四的倍数，QImage就会在每行的后面补齐。所以当第一行的像素点拷贝完成后，它会把mat第二行的像素点拷贝到QImage第一行后面自动补齐的位置，这样第二行图像数据就会比原视频往前偏移，因此导致出现视频是斜着显示，为防止这种情况出现，可将宽高设为4的倍数防止画面倾斜，再设置输出的像素格式为RGB格式，计算出每一帧图片的大小，通过初始化像素格式转换上下文，将像素格式保存在上下文中，接下来使用sws\_scale()将YUV420p格式的视频像素转换成RGB24格式的视频像素，最后将视频数据进行拷贝，防止在读写时候发生冲突，将拷贝后的视频数据以信号形式发送到QWiget中进行图片渲染和接收，核心代码如下：

\_vSwsOutSpec.pixFmt- =AV\_PIX\_FMT\_RGB24

\_vSwsOutSpec.size=av\_image\_get\_buffer\_size()

\_vSwsCtx=sws\_getContext()

onPlayerFrameDecoded()。

视频像素格式转换图，如图5-6所示。

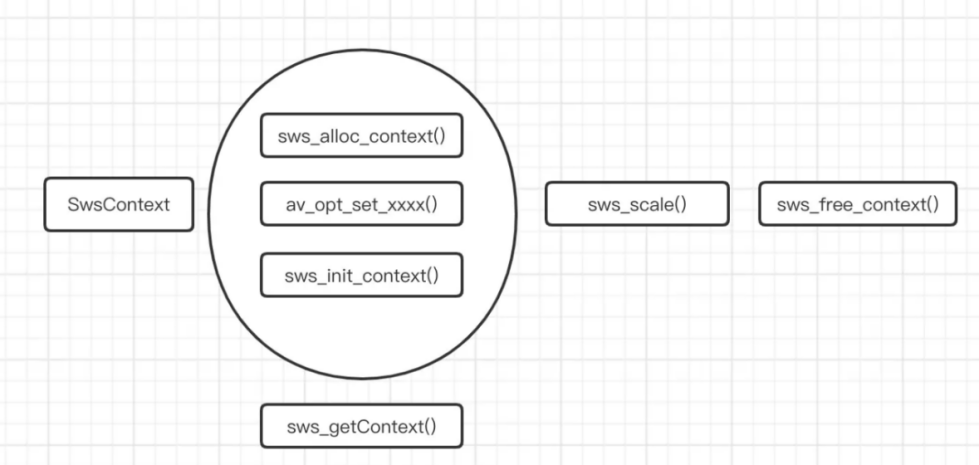


图5-6视频像素格式转换流程图

## 5.4图片绘制模块的详细设计与实现

图片绘制是指将图像绘制到画布或屏幕上的过程。这通常涉及到使用绘图工具或编程语言中的绘图库来创建和操作图像。图片绘制可以实现很多不同的效果，例如图像处理、图形设计、游戏开发、动画制作等。在进行图片绘制时，需要考虑多个因素，例如：

1、图片格式：不同的图片格式（如PNG、JPEG、BMP等）具有不同的特点和优势，需要根据具体需求选择适合的格式。

2、分辨率：高分辨率的图片可以提供更清晰的细节和色彩，但也会占用更多的存储空间和处理资源。

3、色彩空间：不同的色彩空间（如RGB、CMYK、YUV等）提供了不同的颜色表示方式，需要根据具体应用场景选择适合的空间。

4、绘制方式：绘制图片可以通过手工作图或使用计算机程序自动生成，需要根据具体需求选择合适的方式。

5、绘制效果：绘制图片时可以通过调整亮度、对比度、饱和度等参数来实现不同的效果，例如高光、阴影、模糊等。

通过接收到视频解码线程解码出来的图片，通过image=new QImage()在接收到线程发送到的图片后实现对图片资源的创建，将图片发送到 QWidget类中的paintEvent()函数将图片会绘制到幕布上实现图片的显示QPainter(this).drawImage(\_rect,\*\_image)，并且通过onPlayerFrameDecoded（）函数通过对应的图片宽高比实现图片对应的缩放功能，让用户在进行播放时可以随意调节播放窗口大小。

图片渲染显示图，如图5-7所示

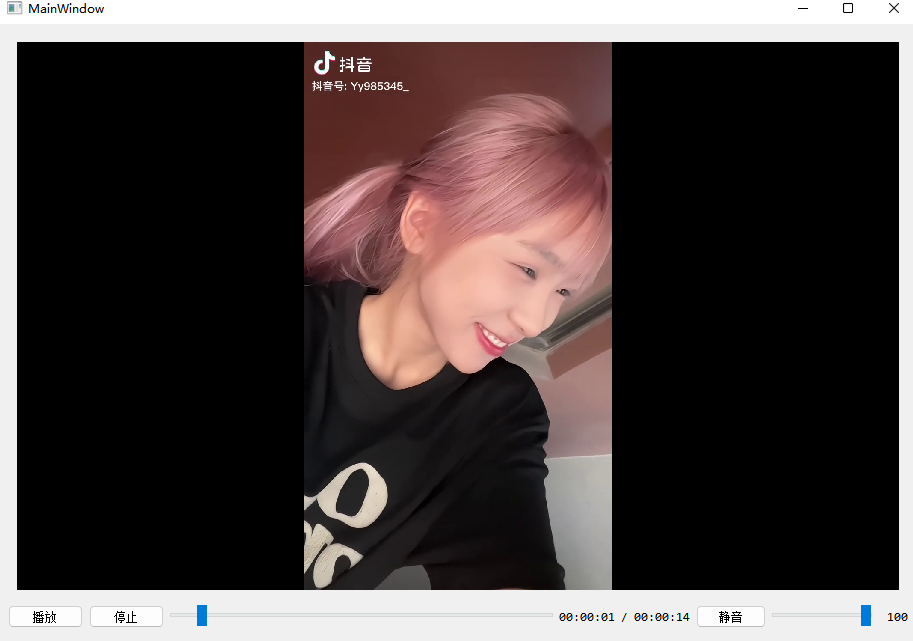


图5-7图片渲染显示图

## 5.5锁和条件变量的详细设计与实现

1、多线程

多线程主要是为了提高系统的性能和响应性。通过利用多核CPU并行执行任务，系统可以更加高效地处理大量的计算和IO操作，从而提高整体的处理速度和吞吐量。此外，多线程还可以使得程序更加灵活，便于设计和维护，并且改善用户体验，提高系统的可用性。因此，在需要支持高并发、高性能、高可用的场景下，采用多线程技术可以是一个不错的选择。

本系统是多线程的为了保证线程安全问题在本系统中实现了锁的应用，因为多线程在访问同一资源是会出现线程安全问题，在同一时刻多线程访问统一资源时如果没用任何安全机制可能会造成系统资源错乱，本系统是采用SDL中的互斥锁是一种并发编程中常用的同步原语，用于控制对共享资源的访问。在多线程环境下，当多个线程尝试同时访问共享资源时，会出现数据竞争等问题，因此需要使用互斥锁来确保同一时间只有一个线程能够访问共享资源，从而避免冲突和不一致性的问题。互斥锁在被一个线程占用时，其他线程必须等待该线程释放锁后才能继续访问共享资源，通过SDL\_mutex \*\_mutex=nullptr定义SDL的互斥锁，再通过调用SDL\_CreateMutex()实现对互斥锁的创建，最后实现加锁和解锁的功能。

2、条件变量

条件变量是一种同步工具，用于协调多个线程对共享资源的访问。条件变量通常与互斥锁配合使用，解决多线程访问共享资源的问题。当一个线程需要等待某个条件满足时，它会在获取互斥锁后检查该条件是否满足。如果条件不满足，该线程会通过条件变量进入等待状态，同时释放互斥锁。当其他线程改变了条件并发出信号时，等待的线程被唤醒并重新获取互斥锁，再次检查条件是否满足。如果条件已满足，线程会释放互斥锁并继续执行，本系统通过条件变量使系统进入生产者和消费者模式，当解封装线程解封装出对应的数据流时，会放入对应的缓冲区在通过条件变量去唤醒对应的解码线程，解码线程获取到互斥锁对流资源进行解码，当条件不满足时系统的音频线程和视频线程将进入堵塞状态等待被唤醒，并且会释放出互斥锁。

3、生产者消费者模型

生产者消费者模型是一种常用的并发编程模型，用于协调多个线程或进程之间的数据交换。在该模型中，有两类线程：生产者和消费者。生产者负责生产数据，并将其放入一个共享的缓冲区中；而消费者则从缓冲区中获取数据并进行处理。因为生产者和消费者可能运行在不同的线程或进程中，因此需要使用同步机制来保证它们之间的正确协作。

在生产者消费者模型中，共享缓冲区可以看作是一个队列。当生产者生产数据时，它将数据插入到队列的末尾；而当消费者需要获取数据时，它从队列的头部取出数据。由于队列是共享资源，因此需要使用互斥锁来保护对队列的访问。同时，在队列为空或已满的情况下，生产者和消费者需要等待对方完成操作，避免死锁或其他并发问题。

为了实现等待通知机制，生产者和消费者还需要使用条件变量。当队列已满时，生产者会通过条件变量进入等待状态，直到有消费者取走了一些数据后唤醒它；而当队列为空时，消费者也会通过条件变量进入等待状态，直到有生产者放入了一些数据后唤醒它。通过互斥锁和条件变量的协作，生产者消费者模型可以实现线程间安全的数据交换，避免了数据竞争和死锁等并发编程问题。

定义出SDL中的条件变量SDL\_cond \*\_cond=nullptr，调用SDL\_CreateCond()去创建条件变量，当条件不满足是是线程进行堵塞状态，并交出互斥锁SDL\_CondWait()，当条件满足是调用SDL\_CondSignal()去唤醒被堵塞的线程，对缓冲区中的资源进行消费。生产者和消费者模型图如图5-8所示。

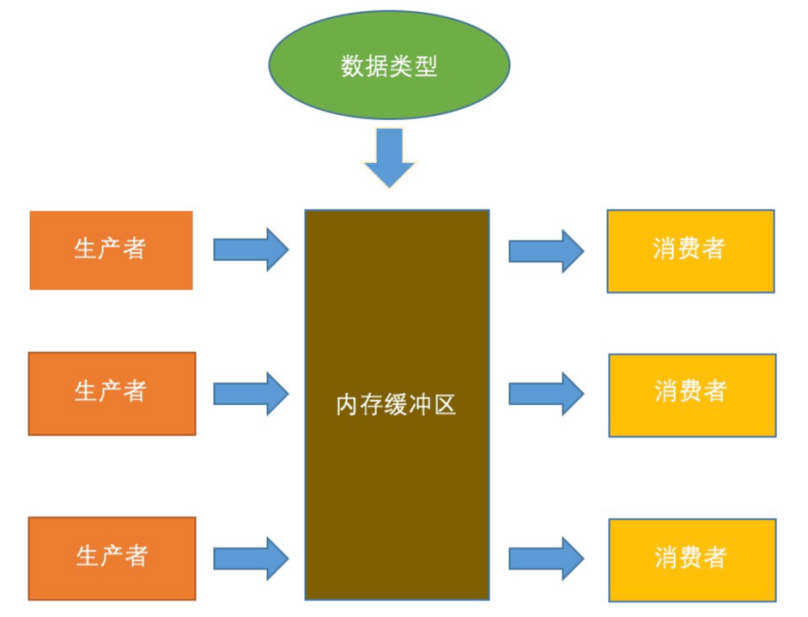


图5-8生产者和消费者模型图

## 5.6音视频文件seek的详细设计与实现

在音视频播放系统中，seek（快进/后退）是指用户通过拖动进度条或按下快进/后退按钮来跳转到视频或音频文件的另一个时间点。这个过程通常需要解码一段新数据，然后将其显示或播放出来。在实现快进/后退功能时，需要考虑以下一些因素：

1、准确性：快进/后退操作需要精确地跳转到指定的时间点，避免出现误差和卡顿。

2、响应速度：快进/后退操作通常需要很快地响应用户的输入，并尽可能快地加载和播放新的数据。

3、可靠性：快进/后退操作可能会导致应用程序崩溃或产生其他异常情况，需要进行充分的测试和错误处理。

4、界面体验：快进/后退操作对用户来说是一种交互体验，需要提供良好的界面反馈和用户引导。

本系统seek的具体实现为：根据用户的输入计算出目标时间点的位置，例如在拖动进度条时可以根据进度条的位置计算出对应的时间点。将目标时间点转换为相应的帧数或采样点数，以便于和实际的音视频帧进行匹配。暂停当前的播放器，并清空缓存中的数据。跳转到目标时间点所对应的帧数或采样点数，开始解码新的数据，并将其存储到缓存中。

在seek操作时应该优先使用音频流进行操作，因为本系统是通过一音频为时间基做同步的，所以在进行操作时应该优先选择音频流，获取到音频流的时间，并转换为时间戳，公式如下：时间戳 = 现实时间 / av\_q2d(时间基)。最后调用av\_seek\_frame()方法实现音视频的seek功能，在对音频进行seek时，要判断音频的播放时间与seek的时间长度对比，因此要进行数据报的释放。在视频seek时应该在解码之后进行判断，原因是视频解码出来的P帧，B帧要依赖I帧去进行解码，所以应该等视频解码完成后再进行对比，如果视频的播放的时间小于seek的时间，将视频包丢弃，在seek成功为防止内存泄漏，需要释放音频数据包和视频数据包。

## 5.7音视频同步详细设计与实现

在音视频播放中，音频和视频通常是分别解码和播放的。由于解码和播放过程的复杂性和不同设备的处理能力差异，可能会导致音频和视频之间出现不同步的情况，即声音和图像没有达到理想的对齐效果。因此，需要进行音视频同步来确保音频和视频播放的一致性。在实现音视频同步时，可以采用以下几种方法：

1、基于时间戳的方法是通过比较音频帧和视频帧的时间戳来确定它们之间的差异，并通过速度控制或插入静帧等方式进行同步。具体而言，如果发现音频帧的时间戳比视频帧的时间戳要晚，则可以暂停视频的播放一段时间，直到音频追上视频；如果音频帧的时间戳比视频帧的时间戳早，则可以快进视频直到与音频同步。

2、基于缓存的方法是将音频和视频数据流缓存到内存中，在两种流之间进行调整以消除延迟并保持同步。具体而言，在播放音频和视频时，可以先将一段数据缓存到内存中，然后在两种流之间进行调整，以确保它们始终保持对齐状态。

3、基于时间轴的方法是在时间轴上将音频和视频进行匹配，确保它们在时间上的位置相同。例如，在播放时可以向视频中插入静帧，以便将视频的时间轴与音频的时间轴对齐。当发现音频和视频之间存在差异时，可以通过调整音频或视频的播放速度来进行同步调整，确保它们始终保持对齐状态。

4、基于外部信号的方法是从外部获得同步信号，例如通过GPS或其他传感器来控制音频和视频的播放，以确保它们始终保持同步状态。这种方法需要依赖外部设备，并且可能会有一定的成本和技术难度。

本系统采用的是基于时间戳做的音视频同步，并且是基于音频的时间做同步，因为在解码出来的PCM数据后音频的采样率是知道的，所以能通过采样率去计算出每个采样点的时间戳，这样音频的时间戳就是相当于固定的，在当解码出视频后，根据音频时间戳和视频时间戳的差异来进行同步处理。当音频时间戳大于视频时间戳时，可以暂停视频播放一段时间直到音频追上视频；当音频时间戳小于视频时间戳时，可以快进视频直到与音频同步。

第6章 音视频编解码技术实现

## 6.1视频编码技术实现

视频编码技术是将视频信号转换为数字信号的过程，以便可以在数字设备上存储、传输和播放视频。视频编码技术旨在通过使用压缩算法来减小视频文件的大小，同时保持高质量的视频画面，常见的编码技术如下。

1、MPEG：MPEG（Moving Picture Experts Group）是一种视频编码标准，它包括多种编码格式，如 MPEG-1、MPEG-2、MPEG-4 和 MPEG-7。这些编码格式可用于不同的应用场景，例如 MPEG-1 用于 VCD，MPEG-2 用于 DVD 和数字电视，MPEG-4 用于流媒体和移动设备。

2、H.264/AVC：H.264/AVC（Advanced Video Coding）是一种广泛使用的视频编码标准，它可以提供高质量的视频压缩，同时保持较低的比特率。H.264/AVC 常用于高清视频和网络视频传输。

3、HEVC/H.265：HEVC（High Efficiency Video Coding）也称为 H.265，是一种新的视频编码标准，它比 H.264/AVC 更先进。HEVC 可以在相同画质下提供更高的压缩率，从而减小视频文件的大小。HEVC 主要用于高清和超高清视频格式。

4、VP9：VP9 是一种由 Google 开发的开放源代码视频编码标准，它可以提供高效的视频压缩，并支持 4K 和 8K 分辨率的视频。VP9 常用于网络视频传输，如 YouTube。

在进行视频播放时，视频的编码方式主要采用H264，H.264是一种视频编码标准，也称为AVC，它是现代视频编码技术中最广泛使用的一种。H.264编码技术可以提供高质量的视频压缩，并且在降低比特率的同时保持高清晰度，因此适用于多种应用场景，如流媒体、视频会议、数字电视和蓝光光盘等，在H264中主要采用的编码技术为：

1、帧内预测：帧内预测是通过对当前帧内的像素进行预测来减少冗余数据的一种技术。在 H.264 编码中，图像被分为若干个小块，每个小块内的像素值可以通过预测相邻的像素值来获得。

2、运动估计：运动估计是通过比较相邻帧之间的像素值来确定运动物体的位置和速度，以减少视频中的冗余数据。在 H.264 编码中，运动估计采用了一种自适应的算法，可以根据视频内容自动调整搜索范围和精度。

3、帧间预测：帧间预测是通过对前一帧或后一帧图像的像素进行预测来减少冗余数据。在 H.264 编码中，帧间预测采用了多种预测模式，如块匹配和运动矢量预测等。

4、变换编码：变换编码是通过将图像从时域转换为频域来减少冗余数据。在 H.264 编码中，变换编码采用了离散余弦变换（DCT）和离散余弦变换（DST）等技术。熵编码：熵编码是通过对变换后的数据进行压缩来减小视频文件的大小。在 H.264 编码中，熵编码采用了 CABAC 和 CAVLC 两种技术。

## 6.2视频解码技术实现

视频解码是将压缩过的视频数据还原为原始视频信号的过程，通常需要通过硬件或软件解码器来完成。以下是视频解码的一般流程：

1、读取视频文件：首先需要读取视频文件，并将其中的视频数据传递给解码器。

2、解压视频数据：在解码器内部，视频数据被解压缩为原始的视频帧数据。通常情况下，视频数据是经过帧间压缩和帧内压缩两个阶段的压缩。

3、解码视频数据：解压缩后的视频数据需要进行解码，以还原为原始的视频信号。解码器通常会使用特定的解码算法，如反量化、反变换等。

4、还原视频帧：在解码视频数据后，需要将其还原为视频帧。视频帧通常包括亮度（Y）和色度（U、V）三个分量。解码器会对这些分量进行重组，以生成完整的视频帧。

5、显示视频帧：最后，解码器将还原后的视频帧传递给显示设备，以在屏幕上显示视频画面。通常情况下，视频解码器会将视频帧缓存一段时间，以确保视频画面的流畅性和稳定性。

本系统通过使用FFmpeg实现对视频的解码功能，首先调用avcodec\_find\_decoder，根据指定的AVCodecID查找注册的解码器，在找到解码器之后要去根据解码器ID初始化相应的裸流解析器，再为解码器上下文分配空间，调用avcodec\_open2，打开解码器，再打开解码器后要从缓冲队列中取出视频的压缩包Packet去发送到解码器中，解码器将Packet视频包解码后，调用avcodec\_receive\_frame，获取到解码后的AVFrame数据，在将视频数据发送到幕布中通过Qt中的paintEvent实现渲染。

视频解码流程图，如图6-1所示

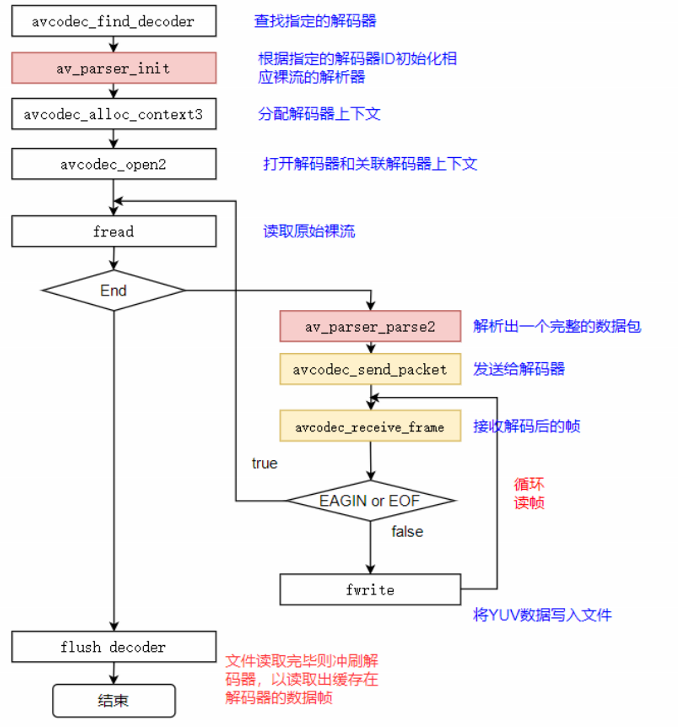


图6-1视频解码流程图

## 6.3音频编码技术实现

音频编码技术是将原始音频信号压缩为更小的数据流的过程，以便于存储和传输。以下是几种常见的音频编码技术：

1、PCM（Pulse Code Modulation）：PCM是最基本的音频编码技术，将原始音频信号转换为数字信号，并采用线性量化技术将数字信号压缩为更小的数据流。PCM编码不会损失任何音频信息，但数据量较大。

2、ADPCM（Adaptive Differential Pulse Code Modulation）：ADPCM是一种改进的PCM编码技术，通过利用音频信号的相关性，将采样的差异进行编码，从而进一步减小数据流的大小。ADPCM编码可以在保持音频质量的同时减少数据量。

3、MP3（MPEG-1 Audio Layer III）：MP3是一种基于互联网流媒体技术的音频压缩格式，采用了一种称为“掩蔽听觉门限”（Masking Auditory Threshold）的技术，可以在保持音质的同时减少数据量。AAC（Advanced Audio Coding）：AAC是一种高效的音频编码技术，采用了一种称为“谱线性预测”（Spectral Band Replication）的技术，可以在减少数据量的同时保持高质量的音频。

4、Vorbis：Vorbis是一种开放源码的音频编码技术，采用了一种称为“有损无损混合”（Hybrid Lossless/Lossy）的技术，可以在保持音频质量的同时减少数据量。

本系统中音频采用的AAC编码。AAC（Advanced Audio Coding）是一种高效的音频编码技术，它采用了一种称为“谱线性预测”（Spectral Band Replication）的技术，可以在减少数据量的同时保持高质量的音频。AAC编码技术是MPEG-4标准中定义的音频编码标准之一，也是很多数字音频设备和互联网音频流媒体服务所采用的音频格式之一，AAC编码主要采用了谱线性预测、码率控制、掩蔽效应。

1、谱线性预测：谱线性预测是一种音频信号压缩技术，它可以将音频信号分成多个频带，然后对不同频带的信号进行不同的压缩。这种技术可以减少数据量，同时保持高质量的音频。谱线性预测在AAC、Vorbis等音频编码技术中广泛应用。

2、码率控制：码率控制是一种自适应技术，它可以根据音频信号的复杂度和编码器的处理能力，自动调整编码比特率，以最大限度地减少数据量，并保持高质量的音频。码率控制在许多音频编码技术中都有应用，例如AAC、MP3等。

3、掩蔽效应：是一种人耳听觉特性，即较大的声音可以掩盖较小的声音。基于这种特性，音频编码器可以在编码过程中将较小的信号掩盖在较大的信号之下，以减少数据量。掩蔽效应在许多音频编码技术中都有应用，例如AAC、MP3等。

## 6.4音频解码技术实现

音频解码技术是将压缩的音频数据流还原为原始音频信号的过程，音频解码步骤主要为：

1、数据解码：将压缩的音频数据流解码为原始PCM数据流。不同的音频编码格式采用不同的压缩算法和解码方式，解码器根据解码规则将压缩数据流解码。

2、重构滤波：将PCM数据流通过重构滤波器进行还原。在音频压缩过程中，为了减少数据量，压缩器通常会对音频信号进行滤波和下采样等处理，解码器需要将这些处理反向执行，通过重构滤波器将PCM数据流还原为原始音频信号。

3、量化还原：将在压缩过程中采用的量化技术还原。在音频压缩过程中，为了减少数据量，压缩器通常会采用量化技术将音频信号进行精度降低，解码器需要将这些处理反向执行，通过还原量化值将PCM数据流还原为原始音频信号。

4、重构重采样：将还原的音频信号通过重构重采样器进行还原。在音频压缩过程中，压缩器通常会对音频信号进行重采样以减少数据量，解码器需要将这些处理反向执行，通过重构重采样器将还原的音频信号进行还原和重构。

5、输出音频信号：将还原的音频信号输出。解码器将还原的音频信号输出到扬声器、耳机或其他音频设备中，用户可以通过这些设备听到还原的音频信号。

本系统通过使用FFmpeg实现对音频的解码功能，通过FFmpeg解码器，将压缩的音频数据包发送到FFmpeg解码器中，通过avcodec\_receive\_frame，获取到解码后的AVFrame数，再解码之后要进行重采样功能后，通过将解码后的数据发送到SDL缓冲区，在SDL缓冲区中实现播放的功能。

第7章 系统测试

## 7.1系统测试分析

系统测试分析是软件测试的一个重要环节，其主要目的是对整个软件系统进行全面测试和分析，以确认系统是否符合规格说明书中所规定的要求和功能，并保证其在实际使用中的可靠性、稳定性、性能等方面达到预期。

系统测试分析通常是在软件开发周期的后期进行，它可以帮助发现和修复潜在的缺陷和问题，从而确保软件的质量和可靠性。在进行系统测试分析时，需要遵循以下步骤：

1、确定测试用例和测试数据：测试用例是指一系列测试步骤和输入数据，用来测试软件系统的各种功能和操作。测试数据是指用于测试的各种输入和输出数据，包括正常数据、异常数据和边界数据等。

2、设计测试方案和测试计划：测试方案是指测试的目标、范围、测试方法和测试策略等。测试计划是指具体的测试任务、时间安排和资源分配等。

3、执行测试和记录测试结果：执行测试时需要按照测试用例和测试数据进行测试，并记录测试结果和发现的问题。测试结果通常包括测试通过率、缺陷数量和缺陷等级等。

4、缺陷跟踪和管理：对于发现的问题，需要进行缺陷跟踪和管理，包括缺陷报告、缺陷修复和缺陷验证等。

5、编写测试报告和总结分析：测试完成后需要编写测试报告，包括测试结果、测试总结和分析等，以便记录测试结果和发现的问题，并为后续的软件开发和维护提供参考。

## 7.2系统测试方法

黑盒测试：黑盒测试是一种测试方法，它基于软件系统的外部行为和功能，测试人员不需要了解软件系统的内部实现细节。黑盒测试可以用来测试系统的功能、性能、可靠性等方面，主要是通过输入输出测试数据来验证系统的正确性和稳定性。

白盒测试：白盒测试是一种测试方法，它基于软件系统的内部实现细节，测试人员需要了解软件系统的代码和内部结构。白盒测试可以用来测试系统的代码质量、性能和安全性等方面，主要是通过测试代码的逻辑和执行路径来验证系统的正确性和稳定性。

灰盒测试：灰盒测试是一种介于黑盒测试和白盒测试之间的测试方法，它既考虑软件系统的外部行为和功能，又考虑其内部实现细节。灰盒测试可以用来测试系统的功能、性能、可靠性和安全性等方面，主要是通过测试输入输出数据和代码执行路径来验证系统的正确性和稳定性。

压力测试：压力测试是一种测试方法，它用于测试软件系统在高负载、高并发和异常情况下的表现。压力测试可以用来测试系统的性能、可靠性和稳定性等方面，主要是通过模拟实际使用情况来验证系统的承载能力和响应速度。

安全测试：安全测试是一种测试方法，它用于测试软件系统的安全性和保密性等方面。安全测试可以用来测试系统的漏洞、攻击和数据泄露等方面，主要是通过测试系统的安全策略和保护机制来验证系统的安全性和稳定性。

## 7.3系统测试用例

现采用黑盒测试的方式，对音视频播放系统进行系统测试，通过对播放功能的测试用例为例，测试方法采用黑盒测试中的等价类划分法和边界值分析法，如表7-1所示。

表7-1 播放功能测试用例表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 项目  名称 | 音视频播放系统 | | 程序版本 | | V0.0.1 |
| 测试  环境 | 硬件环境： 1.七彩虹将星笔记本  2.输入设备：键盘、鼠标  软件环境： 1. 操作系统：Windows11 专业版 CentOS7   1. QT5.14.2 2. [ffmpeg-4.3.2-2021-02-27-full\_build-shared](https://github.com/GyanD/codexffmpeg/releases/download/4.3.2-2021-02-27/ffmpeg-4.3.2-2021-02-27-full_build-shared.7z" \t "_blank) | | | | |
| 功能模块 | 播放功能 | | | | |
| 功能特性 | 播放本地和网络音视频文件，支持播放列表功能和特效设置功能。 | | | | |
| 测试目的 | 测试出该功能模块的缺陷与问题，确保该功能模块稳定安全的运行。 | | | | |
| 预置条件 | 音视频文件已经存储在本地或网络上，播放器已经安装并启动，用户已经登录。 | | | | |
| 用例编号 | 测试步骤 | 预期结果 | | 测试结果 | |
| 001 | 打开播放器，点击打开文件，选择本地音频文件 | 音频文件能够正常播放，声音清晰，没有杂音 | | 与预期结果相同 | |
| 002 | 打开播放器，点击打开文件，选择本地视频文件 | 视频文件能够正常播放，画面清晰，没有花屏和卡顿 | | 与预期结果相同 | |
| 003 | 点击打开文件，输入网络音频文件URL | 音频文件能够正常播放，声音清晰，没有杂音 | | 与预期结果相同 | |

续表7-1

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 测试步骤 | 预期结果 | 测试结果 |
| 004 | 点击打开文件，输入网络视频文件URL | 视频文件能够正常播放，画面清晰，没有花屏和卡顿 | 与预期结果相同 |
| 005 | 点击添加到播放列表，添加本地或网络音频文件 | 音频文件能够正常添加到播放列表中，并能够正常播放 | 与预期结果相同 |
| 006 | 点击添加到播放列表，添加本地或网络视频文件 | 视频文件能够正常添加到播放列表中，并能够正常播放 | 与预期结果相同 |
| 007 | 播放过程中拖动进度条 | 播放器能够正常响应，并能够跳转到对应的时间点继续播放 | 与预期结果相同 |
| 008 | 播放过程中切换音视频文件 | 播放器能够正常响应，并能够切换到对应的音视频文件继续播放 | 与预期结果不同 |
| 009 | 播放过程中调整音量大小 | 播放器能够正常响应，并能够调整音量大小 | 与预期结果相同 |

## 7.4测试分析总结

### 7.4.1 系统缺陷报告分析

通过执行以上测试用例，得出以下的缺陷报告表，如表7-2所示。

表7-2 系统缺陷报告表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 项目编号 | | TD001 | | | 产品名称 | 音视频播放系统 | | |
| 测试人员 | | 荆健威 | | | 测试类型 | 功能性测试 | | |
| 测试时间 | | 2023-4-20 | | | 测试地点 | 大庆市东北石油大学 | | |
| 序号 | 测试项 | | 子测试项 | 问题描述 | | | 缺陷类型 | 回归测试结果 |
| 1 | 播放功能 | | 切换文件 | 没有办法进行文件的切换 | | | 建议 | 通过 |
| 2 | 播放功能 | | 暂停功能 | 用户无法暂停文件的播放，影响用户体验。 | | | 中等 | 通过 |

续表7-2

|  |
| --- |
| 问题类型说明：（1）严重：导致死机或误删信息或程序无法控制；  主要功能未实现或流程缺陷。  （2）中等：一般功能点未实现或流程缺陷；  数据结果错误或界面信息错误。  （3）微小：不影响功能实现的其他缺陷。  （4）建议：建议的改进。 |

### 7.4.2 系统测试覆盖分析

测试用例的设计应该基于用户的使用场景，以完全覆盖需求为目标，确保测试用例能够完全执行且通过。从TD001可以看出，测试用例和测试需求是完全覆盖的关系，即每个需求都需要有一个或多个测试用例进行覆盖。具体的覆盖范围可以参考表7-3。测试用例得到全面执行和批准，表明测试要求的覆盖率达到100%。这也意味着测试用例已经涵盖了所有的测试需求，并且能够全面地覆盖用户的使用场景。

表7-3 系统测试覆盖表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 需求/功能 | 用例个数 | 执行总数 | 未执行 | 未/漏测分析和原因 |
| 功能测试 | 9 | 9 | 0 | 一个功能未通过，原因：没用添加更换文件按钮 |
| 性能测试 | 2 | 2 | 0 | 通过 |
| 兼容性测试 | 1 | 1 | 0 | 通过 |
| 安全性测试 | 1 | 1 | 0 | 通过 |
| 用户界面测试 | 3 | 3 | 0 | 通过 |

在测试过程中，测试用例的执行通过率是一个重要的指标。例如，如果执行了16个测试用例，其中有1个没有通过，则通过率为93.75%。这个指标可以反映出功能性测试用例的效率，即测试用例发现缺陷的能力。除了功能性测试用例之外，还有许多其他类型的测试用例，比如性能测试用例和安全性测试用例。这些测试用例通常采用检查项的方式进行测试，并且未计入测试用例的效率范围内。总之，测试用例的效率是评估测试用例质量的一个重要指标，通过率是其中的一个重要衡量标准。不同类型的测试用例在测试过程中有着不同的作用和效果，需要根据具体的测试目标进行选择和设计。

### 7.4.3 系统存在的问题与建议

兼容性问题：一些音视频格式可能无法在播放器中播放，或者播放效果不佳。建议加强对各种音视频格式的兼容性测试，并及时更新播放器以支持新的音视频格式。

界面设计问题：一些播放器的用户界面设计可能不够直观、简洁、美观，或者不符合用户的使用习惯。建议在设计时考虑用户的使用需求和习惯，并进行用户调研，以提高用户的满意度和体验。

功能不足或过多：一些播放器可能缺乏一些用户需要的功能，或者过多的功能使得用户难以使用。建议在设计时根据用户需求和使用频率，进行功能的删减和添加，并确保简单易用。

性能问题：一些播放器可能存在性能问题，如播放卡顿、占用系统资源过多等。建议进行性能测试，优化播放器的性能，提高播放器的稳定性和流畅度。

安全问题：一些播放器可能存在安全问题，如恶意代码注入、数据泄露等。建议加强安全测试和加密措施，保护用户的隐私和安全。

多平台支持问题：一些播放器可能仅支持某些操作系统，无法满足用户在不同平台上的使用需求。建议加强对多平台的支持，以便用户可以在不同的平台上使用同一款播放器。

结 论

本文基于FFmpeg和Qt实现了一款音视频播放系统，并对编解码算法的原理和实现过程进行了深入的研究，该播放器主要包括以下功能模块：

1. 解码模块：使用FFmpeg解码音视频数据，将其转化为可供播放器使用的格式。
2. 播放模块：使用Qt提供的音视频播放系统控件，将解码后的音视频数据进行播放。
3. UI模块：使用Qt提供的UI控件和界面设计工具，实现自定义的用户界面和功能。
4. 控制模块：实现音视频的控制，例如播放、暂停、停止、快进、快退等操作。
5. 处理模块：使用FFmpeg提供的处理功能，实现音视频的特效处理、转码等操作。

该播放器具有以下优点：

1、支持多种音视频格式：由于FFmpeg支持多种音视频格式，因此基于FFmpeg的音视频播放系统能够处理多种格式的音视频数据。

2、高效快速：FFmpeg是一种高效快速的解决方案，能够在短时间内处理大量的音视频数据。

3、可定制化：Qt提供了良好的可扩展性和可定制性，使得开发者能够根据自己的需求进行定制和扩展。

4、跨平台：Qt支持多种平台，使得开发者能够开发出同时支持多种平台的音视频播放系统。

5、功能丰富：基于FFmpeg和Qt的音视频播放系统能够实现多种功能，例如高清视频播放、实时流媒体播放、多路音频切换和特效处理等。 系统研发主要使用到以下关键技术：

1、FFmpeg开源的音视频处理库：实现解码、编码、转换和处理多种音视频格式，包括但不限于MP4、AVI、FLV、H.264等。FFmpeg的优势在于它非常快速和高效，同时具有广泛的应用和良好的兼容性。

2、Qt跨平台的GUI开发框架：提供了丰富的UI控件和界面设计工具，使得开发者能够轻松实现自定义的用户界面和功能。Qt的优势在于它支持多种平台，包括但不限于Windows、Linux、macOS等，同时提供了良好的可扩展性和可定制性，这使得开发者能够根据自己的需求进行定制和扩展。

参考文献

[1] 龚进明,戴红芳.多媒体资源播放器的设计与开发[J].常熟高专学报,2003, (02).

[2] Zi Hui. Design and Implement of Long-Distance Education System Base on Stream Media Technology[J]. Advanced Materials Research,2014,3181(926-930).

[3] 朱申鸣,周强.多媒体技术在计算机软件中的应用分析[J].信息系统工程,2023,No.350(02):66-68.

[4] 周澳蕾.数字媒体技术的应用[J].数字通信世界,2017,No.154(10):37+216.

[5] Bai Yan. Design and Implementation of Music Teaching based on Streaming Media Technology[A]. Institute of Management Science and Industrial Engineering.Proceedings of 2019 4th International Workshop on Materials Engineering and Computer Sciences(IWMECS 2019)[C].Institute of Management Science and Industrial Engineering:计算机科学与电子技术国际学会(Computer Science and Electronic Technology International Society),2019:6.

[6] Zhu Feng Qiao,Jian Xin Guo,Ji Chun Zhao. Research and Application of Two-Dimensional Code in Distance Education Stream Media System[J]. Applied Mechanics and Materials,2013,2700(411-414).

[7] 戴明龙. 多接口智能终端的实时流媒体传输技术优化[D].北京邮电大学,2019.

[8]黄沓锋.流媒体传输加密技术研究[J].中小企业管理与科技(下旬刊),2016(12):189-190.

[9] 韩全生. 基于H.264的音视频解码播放软件设计与实现[D].西安电子科技大学,2012.

[10]代文姣. 基于FFMpeg的稳定应用层组播流媒体直播系统研究[D].华中师范大学,2018.

[11]李胜辉,王文敏.通信音视频编解码技术的研究与应用[J].微型电脑应用,2022,38(11):202-205.

[12]贺良凯.“直播+电商”模式如何推动我国经济的发展[J].中国商论,2022,No.860(13):18-20.DOI:10.19699/j.cnki.issn2096-0298.2022.13.018.

[13]晶晶,刘光尧,汪磊等.FFmpeg在视频图像处理中的应用[J].刑事技术,2020,45(03):234-237.DOI:10.16467/j.1008-3650.2020.03.003.

[14]. 中国实时音视频行业研究报告[C]//上海艾瑞市场咨询有限公司.艾瑞咨询系列研究报告（2022年第8期）.[出版者不详],2022:54.

[15]邓正良.基于FFmpeg和SDL的视频流播放存储研究综述[J].现代计算机,2019,No.658(22):47-50.

致 谢

时光荏苒，匆匆而过的四年大学生活即将画上句号，我心中不禁涌起了深深的感慨和不舍。四年前，我怀揣着梦想和热情，踏入了校园，而如今，我已经要离开这个让我收获满满的地方，前往未知的新征程。回首往事，我感慨万千，内心充满了感激和感恩。

在这四年的成长道路上，我遇到了许多支持我的人，他们给我无私的帮助和支持，让我在迷茫和困惑中找到了方向和信心。在此，我要向他们表达我的谢意和敬意。

首先，我要感谢我的指导老师杜娟在我毕业设计的过程中，她给予了我无微不至的指导和关怀。她不仅帮助我理清了思路，还为我提供了许多宝贵的建议和意见，使我的毕业设计顺利完成。在此，我深深地感谢她对我的支持和帮助，也希望她的工作和生活一切顺利。

其次，我要感谢我的室友们。四年的大学时光，他们一直陪伴在我身边，给我带来了无尽的欢笑和温暖。在宿舍里，我们一起经历了快乐、悲伤、困惑和挑战，共同度过了一段难忘的时光。在此，我要对他们的友谊和支持表示感谢，也祝愿他们在未来的生活中一切都好。

最后，我要感谢我的朋友们。在大学期间，我结识了许多志同道合的朋友，他们在我生活和学习中给我提供了无限的帮助和支持。在这里，我要向他们表示感激和敬意，同时也希望我们的友谊能够持续下去。

离别虽然让人难过，但是我们也要勇敢面对未来的挑战和机遇。我相信，我们会带着刚来大学时的那份向往在未来道路上披荆斩棘，迎来我们自己的春天并且我们每个人都会在自己的道路上越走越远，创造属于自己的美好生活。最后，我祝愿所有的老师和同学们身体健康、万事如意，也希望未来的日子里，我们都能够保持联系，共同见证彼此的成长和进步。