

中科院声学所离线FFMPEG

概要设计说明

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| V1.0 | 王超锋 |  | 2021-11-17 | 创建文件 |
|  |  |  |  |  |

目录

[1.概述 3](#_Toc16808)

[1.1简介 3](#_Toc16401)

[1.2 读者对象 3](#_Toc28535)

[1.3术语定义 4](#_Toc19604)

[2.总体设计 5](#_Toc29126)

[2.1 总体需求 5](#_Toc4418)

[2.2 运行环境 6](#_Toc30597)

[2.2.1 软件环境 6](#_Toc24781)

[2.2.2 硬件环境 6](#_Toc21040)

[2.3系统总体结构设计 7](#_Toc12804)

[3.模块设计 9](#_Toc893)

[3.1ffmpeg主要组成部分 9](#_Toc21860)

[4.测试设计 15](#_Toc298)

[4.1 配置文件设计 15](#_Toc31773)

[4.2 测试功能点 16](#_Toc10339)

[5.可靠性设计 17](#_Toc17503)

[6.扩展性设计 18](#_Toc25966)

[7.维护性设计 19](#_Toc23787)

[8.易用性设计 20](#_Toc31083)

# 1.概述

1

**1.1简介**

FFmpeg是一套可以用来记录、转换数字音频、视频，并能将其转化为流的开源计算机程序。采用LGPL或GPL许可证。它提供了录制、转换以及流化音视频的完整解决方案。

## 1.2 读者对象

本文档的读者对象为离线识别系统的开发人员、测试人员、系统维护人员及接入识别系统的第三方业务人员，通过本文档能够从总体上了解FFMPEG的架构形式及数据流向。

本说明给出FFMPEG的设计说明，包括最终实现的系统必须满足的功能、性能、接口、附属测试工具程序及设计约束等。

目的在于：

* 为开发人员提供依据；
* 为代码修改、维护提供条件；

## 1.3术语定义

语音识别（Automatic Speech Recognition），简称（ASR）。

# 2.总体设计

2

## 2.1 总体需求

功能需求：对语音进行转码；

性能需求：转码耗时低；

接口需求：接口参数简单明了，接口中的每个参数都要有实际意义，保证接口调用流程清晰。

系统验证工具：提供系统完整性，功能正确性的验证工具；

## 2.2 运行环境

### 2.2.1 软件环境

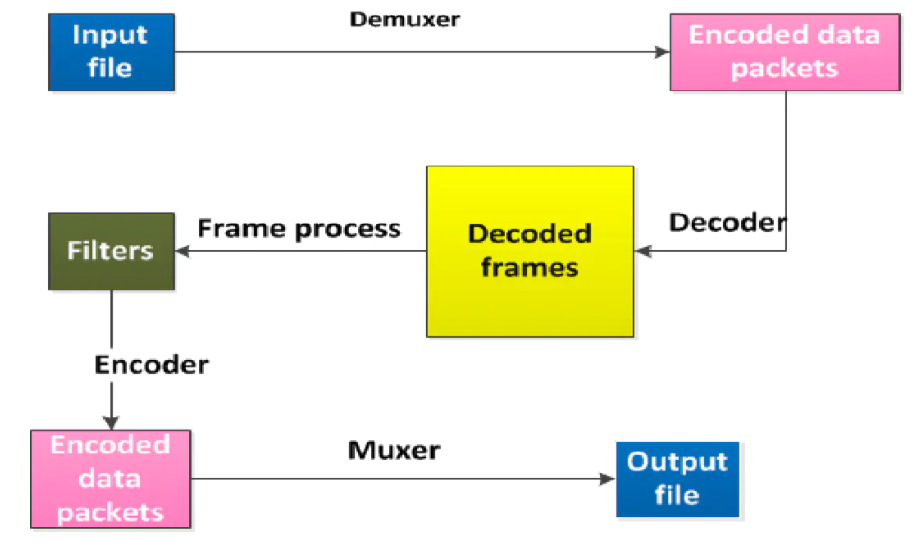
|  |  |  |
| --- | --- | --- |
| 分类 | 名称 | 版本 |
| 操作系统 | Centos | 7.0以上 |
| 数据库 | Redis | 3.2.0 |
| 能力接口 | Tomcat | 9.0.35 |

### 2.2.2 硬件环境

|  |  |  |
| --- | --- | --- |
| 服务器 | 最低配置 | 推荐配置 |
| redis服务器 | CPU：1  内存：256M | CPU：1  内存：1G |
| 识别服务器 | CPU：4  内存：10G  磁盘：20G | CPU：10  内存：15G  磁盘：30G |

## 2.3系统总体结构设计

ffmpeg整个流程如下图所示。



1、FFmpeg程序把-i参数指定的若干文件内容读入到内存，按照输入的参数或者程序默认的参数来处理并且把结果写入到若干的文件中。输入和输出文件可以是计算机文件、管道、网络流、捕获设备等。

2、FFmpeg用libavformat包调用解复用器（demuxers）来读取输入文件中被编码的数据包(packets)，如果有多个输入文件，FFmpeg以有效输入流的最小时间戳来同步，

3、然后解码器（decoder）从已编码的数据包中产生未被压缩的帧（frame），在那之后调用可选的过滤器。

4、这些帧被传递到编码器，编码器会产生新的编码包

5、把新的编码包传递给复用器(muxer)处理并且把结果写入到输出文件中。

# 3.模块设计

3

## 3.1ffmpeg主要组成部分

1、libavformat：用于各种音视频封装格式的生成和解析，包括获取解码所需信息以生成解码上下文结构和读取音视频帧等功能，包含demuxers和muxer库；

2、libavcodec：用于各种类型声音/图像编解码；

3、libavutil：包含一些公共的工具函数；

4、libswscale：用于视频场景比例缩放、色彩映射转换；

5、libpostproc：用于后期效果处理；

6、ffmpeg：是一个命令行工具，用来对视频文件转换格式，也支持对电视卡实时编码；

7、ffsever：是一个HTTP多媒体实时广播流服务器，支持时光平移；

8、ffplay：是一个简单的播放器，使用ffmpeg 库解析和解码，通过SDL显示；

在这组成部分中，需要熟悉基础概念有

容器(Container)

容器就是一种文件格式，比如flv，mkv等。包含下面5种流以及文件头信息。

流(Stream)

是一种视频数据信息的传输方式，5种流：音频，视频，字幕，附件，数据。

帧(Frame)

帧代表一幅静止的图像，分为I帧，P帧，B帧。

编解码器(Codec)

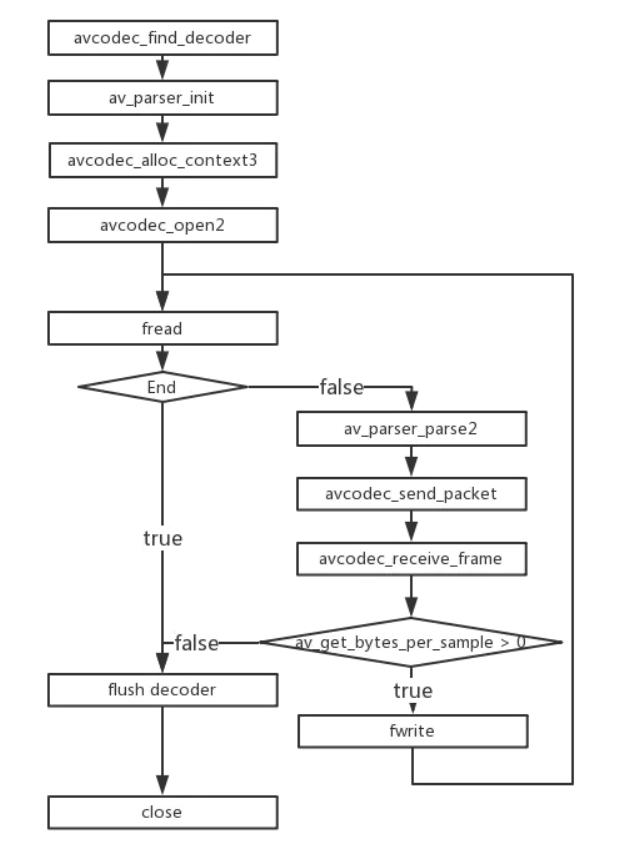
是对视频进行压缩或者解压缩，CODEC =COde （编码） +DECode（解码）

复用/解复用(mux/demux)

把不同的流按照某种容器的规则放入容器，这种行为叫做复用（mux）

把不同的流从某种容器中解析出来，这种行为叫做解复用(demux)

ffmpeg具体处理流程如下：



主要函数介绍：

函数原型：

AVCodec \*avcodec\_find\_decoder(enum AVCodecID id);

参数说明：enum AVCodecID id 解码器的ID

返回值：注册的解码器

功能：

根据指定的AVCodecID查找注册的解码器。

函数原型：

AVCodecParserContext \*av\_parser\_init(int codec\_id)；

参数说明：int codec\_id 解码器的ID

返回值：AVCodecParserContext指针

功能：

初始化AVCodecParserContext。

函数原型：

AVCodecContext \*avcode\_alloc\_context3(const AVCodec \*codec)；

参数说明：AVCodec \*codec 解码器

返回值：AVCodecParserContext指针

功能：

为AVCodecContext分配内存。

函数原型：

int avcodec\_open2(AVCodecContext \*avctx, const AVCodec \*codec, AVDictionary \*\*options);

参数说明：

AVCodecContext \*avctx 需要初始化的AVCodecContext

const AVCodec \*codec 输入的AVCodec

AVDictionary \*\*options 一些选项。例如使用libx264编码的时候，“preset”，“tune”等都可以通过该参数设置

返回值：

int类型。取值及说明如下：

0 ：成功

-1 ：失败

功能：

打开解码器。

函数原型：

int av\_parser\_parse2()；

参数说明：

返回值：

int类型。取值及说明如下：

0 ：成功

<0 ：失败

功能：

解析获得一个Packet。

函数原型：

int avcodec\_send\_packet(AVCodecContext \*avctx, const AVPacket \*avpkt);

参数说明：

AVCodecContext \*avctx 需要初始化的AVCodecContext

const AVPacket \*avpkt 输入的AVPacket

返回值：

int类型。取值及说明如下：

0 ：成功

<0 ：失败

功能：

将AVPacket压缩数据给解码器。

函数原型：

int avcodec\_receive\_frame(AVCodecContext \*avctx, AVFrame \*frame);

参数说明：

AVCodecContext \*avctx 需要初始化的AVCodecContext

AVFrame \*frame 输入的AVFrame

返回值：

int类型。取值及说明如下：

0 ：成功

<0 ：失败

功能：

获取到解码后的AVFrame数据。

函数原型：

int av\_get\_bytes\_per\_sample(enum AVSampleFormat sample\_fmt);

参数说明：

enum AVSampleFormat sample\_fmt sample\_fmt入参

返回值：

int类型。取值及说明如下：

0 ：成功

<0 ：失败

功能：

获取每个sample中的字节数。

关键数据结构说明：

AVCodecParser：用于解析输入的数据流并把它分成一帧一帧的压缩编码数据。比较形象的说法就是把长长的一段连续的数据“切割”成一段段的数据。

# 4.测试设计

4

## 4.1 配置文件设计

设置配置文件，方便使用不同控制逻辑，支持不同的功能及结果输出，配置参数实现如下目的：

* 通过参数控制支持特殊功能

配置主要参数如下表：

|  |  |  |
| --- | --- | --- |
| 参数 | 取值 | 说明 |
| inwav | 字符串 | 待转码语音 |
| -ar |  | 设定声音采样率 |
| -ac |  | 设定声道数 |
| -ab |  | 设定声道数 |
| -f |  | 输出格式 |
| outwav | 字符串 | 转码后语音 |

## 4.2 测试功能点

根据设计需求，对服务必须满足的功能点进行验证测试。

|  |  |
| --- | --- |
| 测试功能 | 期望结果 |
| 使用不同的语音进行转码 | 正确配置参数，转码后语音正常。 |

# 5.可靠性设计

5

离线系统FFMPEG设计考虑了实际应用场景的复杂性和输入的多样性的情况，内部加入了大部分实际情况中会出现的异常情况的处理方案，保证稳定性和可靠性。

# 6.扩展性设计

6

系统本身遵循模块化设计，可以随时添加定制化功能，满足不同应用场景的需要。

# 7.维护性设计

7

离线系统为了方便后期的引擎维护，采用标准c/c++函数同时在较低编译器上进行编译

# 8.易用性设计

8

系统安装部署时有一键安装部署脚本，只需要执行脚本即可安装,减少操作量。系统启停均有对应的脚本，实现自动启动、停止服务。