

中科院声学所离线总控端

数据结构设计说明

目录

[1.概述 5](#_Toc1440)

[1.1简介 5](#_Toc27338)

[1.2 读者对象 5](#_Toc3584)

[3. 数据结构设计 6](#_Toc19763)

[3.1任务消息：TaskInfo 6](#_Toc29258)

[3.2结果消息：ResultInfo 7](#_Toc871)

[3.3 总控端类及结构体定义 8](#_Toc5530)

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| V1.0 | 王超锋 |  | 20211102 |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 1.概述

1

1.1简介

离线系统工程部分主要功能：处理能力接口接收客户端发送的任务。总控端从redis队列里阻塞取出任务，对原始语音进行转码、任务信息入库。任务分别发往识别引擎、聚类引擎、叠音引擎、ITN引擎、标点引擎。

## 1.2 读者对象

本文档是对离线工程部分的内部代码流程和相关技术点的整体说明，因此读者最好满足如下几点条件后再读此文档：

* 有一定的C和C++语言基础

# 3. 数据结构设计

3

本节主要介绍protobuf消息结构设计和XML模块结构体。

总控端和各服务端之间通信的消息结构的封装采用Google Protocol Buffer( 简称 Protobuf)，类TaskInfo用于表示语音任务，类ResultInfo用于表示处理结果。

### **3.1任务消息：TaskInfo**

message TaskInfo

{

//音频信息

required string serialNum = 1;

required string audioname=2;

required string audiourl = 3;

required int32 channels=4;

required string language=5;

optional string keywordlist = 6;

optional string resultfilePath = 7 ;

optional string xmlfilePath=8;

}

说明：

* serialNum：音频文件的唯一性标识。
* audioname：音频文件的名称，例如：1.V3。
* audiourl ：音频的存放目录，例如：/home/thinkit/tempvoice

根据2,3的拼接得到语音文件的绝对路径。

* channels：音频的声道信息，0：单声道；1:双声道。
* language：音频语种信息，zhn：中文，eng：英语 .......。

若未设置，默认为zhn。

* keywordlist ：关键词列表的绝对路径。

若未设置，则默认使用XML服务端目录下的关键词列表。

* resultfilePath ：结果文本文件存放的绝对目录。

若未设置，则默认与xmlfilePath相同。

* xmlfilePath:最终的xml结果文件存放的绝对路径。

请务必给定该路径，否则任务将处理失败。

总控端从数据库中获取每条语音任务的详细信息后，都将对应生成一个个TaskInfo。当语音任务需要发送给某服务端前，再将TaskInfo序列化为一个字符串，发送给服务端进行处理。

### **3.2结果消息：ResultInfo**

message ResultInfo

{

required string serialNum = 1;

required int32 errorCode = 2;

optional bytes audioname=3;

optional bytes xmlfilePath = 4;

}

说明：

* serialNum：音频文件的唯一性标识。与TaskInfo中serialNum对应。
* errorCode：结果错误码。 0表示正常返回，-1表示分析过程有错误
* audioname：音频文件的名称。与TaskInfo中audioname对应。
* xmlfilePath: xml结果存放路径。与TaskInfo中xmlfilePath对应。

各服务端每处理完一个TaskInfo任务后，将相应构建一个ResultInfo，当需要将结果反馈给总控端前，将ResultInfo序列化为一个字符串，发送给总控端进行后续处理。

### 3.3 总控端类及结构体定义

本节主要整理总控端模块的类。

每个服务端都会有一个任务发送线程、结果接收线程和结果处理线程，在开启线程的时候会将相应服务端的信息：服务端名字、通信套接字、任务队列、结果队列，发送给线程，同时将该服务端的下一个服务端也会发送过去，相关结构体为：ServerBaseInfo、SRVMANAGE、ThreadParam。另外，语音编码格式通过枚举类型VoiceType表示。

服务端基本信息结构体：ServerBaseInfo

struct ServerBaseInfo

{

SOCKET consk;

string srvName;

string ip;

int port;

CRITICAL\_SECTION mutex\_update\_socket;

bool isSocketOk;

};

数据成员解释如下：

consk：总控端与该服务端通信的socket。

srvName：服务端名称。

ip：服务端ip。

port：服务端口号。

mutex\_update\_socket：consk的互斥锁。

isSocketOk：consk的连接状态。

服务端信息结构：SRVMANAGE

struct SRVMANAGE

{

int socket\_try\_times;

ServerBaseInfo srvInfo;

myQueue<TaskInfo\*> tskQ;

myQueue<ResultInfo\*> resQ;

SRVMANAGE()

{

socket\_try\_times = 0;

}

};

数据成员解释如下：

socket\_try\_times为总控端与该服务端通信的socket连接异常时，已经尝试重连的次数。

tskQ：该服务端的任务队列。TaskInfo结构体定义参见3.1。

resQ：该服务端的结果队列。ResultInfo结构体定义参见3.2。

srvInfo: 该服务端信息。

线程参数结构体：ThreadParam

struct ThreadParam

{

SRVMANAGE\* curSrv;

SRVMANAGE\* nextSrv;

ThreadParam()

{

curSrv= nextSrv = NULL;

}

};

数据成员解释如下：

curSrv：当前的服务端。

nextSrv：下一个服务端。

语音编码格式枚举类型：VoiceType

enum VoiceType

{

sum=0, //多种格式混合语音

pcm=1, //128kbps的pcm: 8k\_16bit\_PCM

vox=3, //6k\_4bit的vox: 6k\_4bit\_Vox

acm=6, //adpcm

mp3=8, //MP3语音: 44100Hz\_16bit\_128kbps\_mp3

};