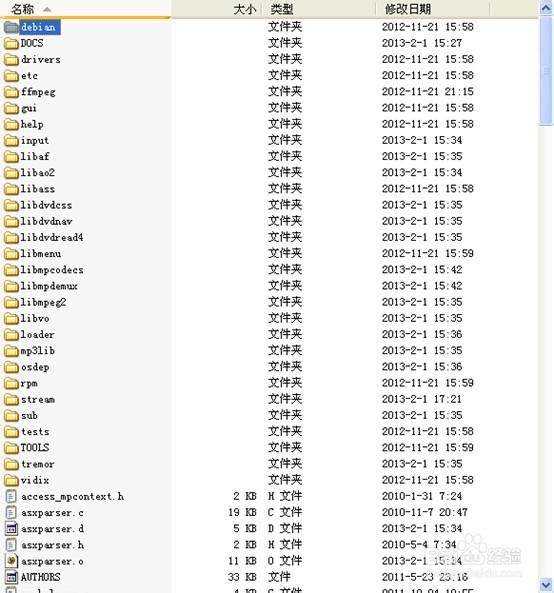
MPlayer源码走读

MPlayer源码结构如下



所有宏定义在config.h中

(1)播放器处理之前的初始化  common\_preinit();包含有计时器、消息系统等的初始化

(2)创建配置的上下文 并注册配置项  mconfig = m\_config\_new();

根据输入参数判断是否是GUI 或者NOGUI

parse\_cfgfiles(mconfig);解析配置文件

(3)通用设置  common\_init() 包括路径设置set\_path\_env();解析codec配置寄存器

parse\_codec\_cfg(NULL);

(4)GUI的相关设置 guiPlaylistInitialize(mpctx->playtree, mconfig, enqueue);

(5)一些提示信息的打印

(6)判断是否定义有RTC和GUI

(7)键盘输入的初始化 mp\_input\_init();

(8)加载配置文件load\_profile\_config(mconfig, filename);

(9)分析播放树  while (player\_idle\_mode && !filename) {……}

两种模式 player\_idle\_mode (播放空闲模式)和  slave\_mode (后台播放模式)

(10) 打开播放的流 mpctx->stream  = open\_stream(filename, 0, &mpctx->file\_format);

进入打开流的核心方法 (Stream.c)

stream\_t\* open\_stream\_full(const char\* filename,int mode, char\*\* options, int\* file\_format)

在这个方法中有个重要的 数组 const stream\_info\_t\* const auto\_open\_streams[] 定义了流的各种不同类型

通过比对Config.h中宏定义   定义了 CONFIG\_NETWORKING、CONFIG\_TV、CONFIG\_FTP 说明了mplayer的流支持了这些协议

取其中的一个函数 &stream\_info\_http1  分析http1的流是怎样打开并且回调的

const stream\_info\_t stream\_info\_http1 = {

  "http streaming",   //定义流信息 info

  "null",  //定义 流的name

  "Bertrand, Albeau, Reimar Doeffinger, Arpi?",  //定义流 author

  "plain http",  //定义这个流的相关注释

 open\_s1,

  {"http", "http\_proxy", "unsv", "icyx", "noicyx", NULL},  //定义这个流支持的协议

  NULL,

  0 // Urls are an option string

};

主要的函数是流的打开回调接口  open\_s1   (里面定义了流的open 、close 等等操作)

通过 file\_format 可以与后面我们的demux模块对应

如果是rtsp的流媒体 就会走下去stream\_info\_rtsp  通过打开rtsp流的回调函数

rtsp\_streaming\_open走到rtsp的网络请求中去。rtsp\_streaming\_start 中

connect2Server (stream->streaming\_ctrl->url->hostname,

port = (stream->streaming\_ctrl->url->port ?stream->streaming\_ctrl->url->port :RTSP\_DEFAULT\_PORT), 1);

connect2Server\_with\_af中具体的根据ipv4或者ipv6请求网络连接

在这里经常会遇到给的流无法处理，首先需要查看auto\_open\_streams[] 中是否定义了支持该协议，并且在Config.h中是否定义了支持该协议的宏；如果得到的是网络流  还要检查网络是否连接正常

(11) 申请播放的缓存 stream\_enable\_cache

(12) 分离音视频流 demux\_open(mpctx->stream, mpctx->file\_format, audio\_id, video\_id, dvdsub\_id, filename)

demux\_open\_stream(vs, demuxer\_type ? demuxer\_type : file\_format,

   demuxer\_force, audio\_stream ? -2 : audio\_id,video\_id, sub\_stream ? -2 : dvdsub\_id, filename);

在这个函数中有个函数get\_demuxer\_desc\_from\_type(file\_format) 定义了 分离器的类型 const demuxer\_desc\_t \*const demuxer\_list[]

同打开流媒体一样我们比对Config.h中定义的codec libraries ，以&demuxer\_desc\_asf为例

const demuxer\_desc\_t demuxer\_desc\_asf = {

  "ASF demuxer",  //定义分离器的信息 一般是别名或者描述

  "asf",  //分离器的name

  "ASF", //分离器探测到时候打印的描述信息

  "A'rpi", //分离器的author

  "ASF, WMV, WMA",    //分离器介绍

  DEMUXER\_TYPE\_ASF,  //分离器类型

  1, // safe autodetect   //置为1表示在文件扩展名之前检查

  asf\_check\_header,    //用来检查是否可以分离这个file  如果可以分离返回分离器的类型

  demux\_asf\_fill\_buffer,

  demux\_open\_asf,

  demux\_close\_asf,

  demux\_seek\_asf,

  demux\_asf\_control

}

上面定义的6个回调函数分别是：检查、解封装、打开、关闭、拖动、控制，大都是可选的，如果觉得没有用设置为NULL

将分离得到的buffer 用不同的流封装

mpctx->d\_audio = mpctx->demuxer->audio;

mpctx->d\_video = mpctx->demuxer->video;

mpctx->d\_sub   = mpctx->demuxer->sub;