# Android 的多媒体框架 OpenCore 介绍

分为几个阶段：

1、整个 Android 的多媒体框架 OpenCore

2、Player 和 Author 的详细介绍

2、OpenCore 和 Android 其他部分的集成

## 第一部分 OpenCore 概述

OpenCore 的另外一个常用的称呼是 PacketVideo，它是 Android 的多媒体核心。在防站的过程中，PacketVideo 是一家公司的名称，而 OpenCore 是这套多媒体框架的软件层的名称。在 Android 的开发者中间，二者的含义基本相同。对比 Android 的其它程序库，OpenCore 的代码非常庞大，它是一个基于 C++的实现，定义了全功能的操作系统移植层，各种基本的功能均被封装成类的形式，各层次之间的接口多使用继承等方式。

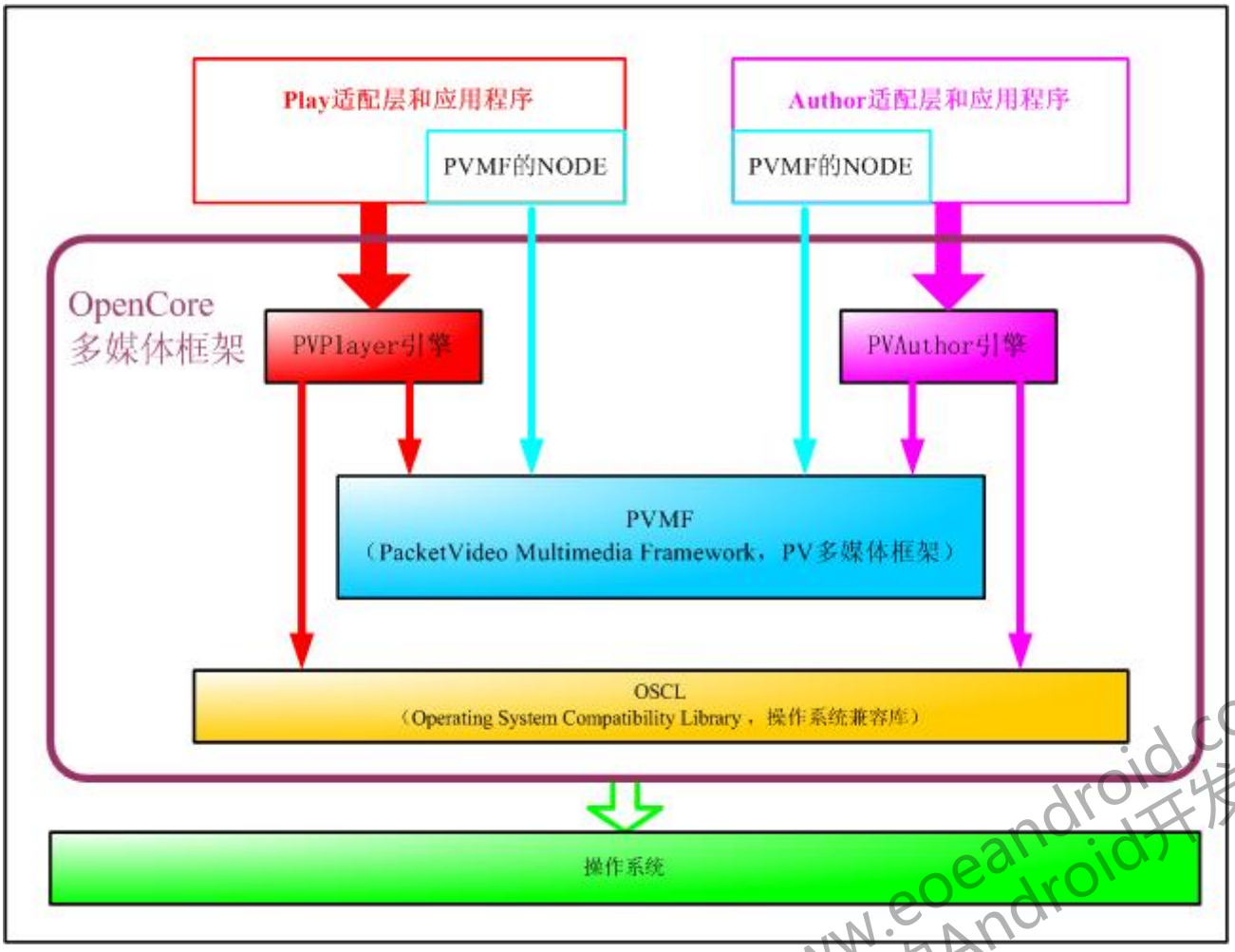
OpenCore 是一个多媒体的框架，从宏观上来看，它主要包含了两大方面的内容：

PVPlayer：提供媒体播放器的功能，完成各种音频（Audio）、视频（Video）流的回放（Playback）功能

PVAuthor：提供媒体流记录的功能，完成各种音频（Audio）、视频（Video）流的以及静态图像捕获功能

PVPlayer 和 PVAuthor 以 SDK 的形式提供给开发者，可以在这个 SDK 之上构建多种应用程序和服务。在移动终端中常常使用的多媒体应用程序，例如媒体播放器、照相机、录像机、录音机等等。

为了更好的组织整体的架构，OpenCore 在软件层次在宏观上分成几个层次：




OSCL：Operating System Compatibility Library (操作系统兼容库)，包含了一些操作系统底层

的操作，为了更好地在不同操作系统移植。包含了基本数据类型、配置、字符串工具、IO、错误处理、线程等内 容，类似一个基础的C++库。

PVMF：PacketVideo Multimedia Framework（PV多媒体框架），在框架内实现一个文件解析（parser）和组成（composer）、编解码的NODE，也可以继承其通用的接口，在用户层实现一些NODE。

PVPlayer Engine：PVPlayer引擎。

PVAuthor Engine：PVAuthor引擎。

事实上，OpenCore中包含的内容非常多：从播放的角度，PVPlayer的输入的（Source）是文件或者网络媒体流，输出（Sink）是 音频视频的输出设备，其基本功能包含了媒体流控制、文件解析、音频视频流的解码（Decode）等方面的内容。除了从文件中播放媒体文件之外，还包含了与 网络相关的RTSP流（Real Time Stream Protocol,实时流协议）。在媒体流记录的方面，PVAuthor的输入的（Source）是照相机、麦克风等设备，输出（Sink）是各种文件， 包含了流的同步、音频视频流的编码（Encode）以及文件的写入等功能。

在使用OpenCore的SDK的时候，有可能需要在应用程序层实现一个适配器（Adaptor），然后在适配器之上实现具体的功能，对于PVMF的NODE也可以基于通用的接口，在上层实现，以插件的形式使用。

# 第二部分 OpenCore的代码结构

## 2.1 代码结构

以开源Android的代码为例，OpenCore的代码在以下目录中：external/opencore/。这个目录是OpenCore的根目录，其中包含的子目录如下所示：

android：这里面是一个上层的库，它基于PVPlayer和PVAuthor的SDK实现了一个为Android使用的Player和Author。

baselibs：包含数据结构和线程安全等内容的底层库。

codecs_v2：这是一个内容较多的库，主要包含编解码的实现，以及一个OpenMAX的实现。

engines：包含 PVPlayer和 PVAuthor引擎的实现。

extern_libs_v2：包含了khronos的OpenMAX的头文件。

fileformats：文件格式的解析（parser）工具。

nodes：提供一些PVMF的NODE，主要是编解码和文件解析方面的。

oscl：操作系统兼容库。

pvmi： 输入输出控制的抽象接口。

protocols：主要是与网络相关的RTSP、RTP、HTTP等协议的相关内容。

pvcommon：pvcommon库文件的Android.mk文件，没有源文件。

pvplayer：pvplayer库文件的Android.mk文件，没有源文件。

pvauthor：pvauthor库文件的Android.mk文件，没有源文件。

tools_v2：编译工具以及一些可注册的模块。

在external/opencore/目录中还有2个文件，如下所示：

Android.mk：全局的编译文件

pvplayer.conf：配置文件

在external/opencore/的各个子文件夹中包含了众多的Android.mk文件，它们之间还存在着“递归”的关系。例如根目录下的Android.mk，就包含了如下的内容片断：

```
include $(PV_TOP)/pvcommon/Android.mk
include $(PV_TOP)/pvplayer/Android.mk
include $(PV_TOP)/pvauthor/Android.mk
```

这表示了要引用pvcommon，pvplayer和pvauthor等文件夹下面的Android.mk文件。

external/opencore/的各个Android.mk文件可以按照排列组合进行使用，将几个Android.mk内容

合并在一个库当中。

## 2.2 编译结构

### 库的层次关系

在 **Android** 的开源版本中编译出来的内容，**OpenCore** 编译出来的各个库如下所示：

**libopencoreauthor.so**：**OpenCore** 的 **Author** 库

**libopencorecommon.so**：**OpenCore** 底层的公共库

**libopencoredownloadreg.so**：下载注册库

**libopencoredownload.so**：下载功能实现库

**libopencoremp4reg.so**：**MP4** 注册库

**libopencoremp4.so**：**MP4** 功能实现库

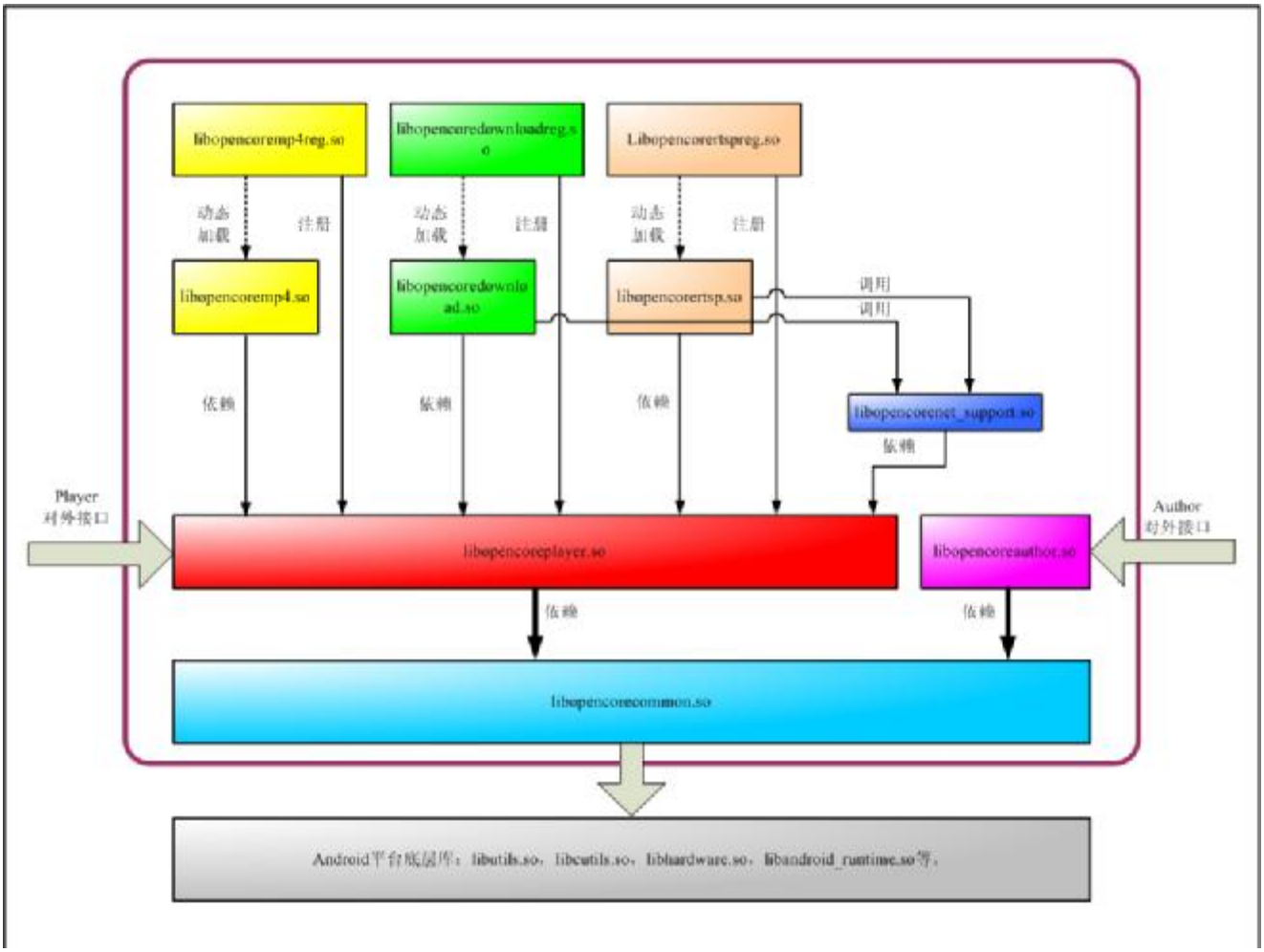
**libopencorenet_support.so**：网络支持库

**libopencoreplayer.so**：**OpenCore** 的 **Player** 库

**libopencorertspreg.so**：**RTSP** 注册库

**libopencorertsp.so**：**RTSP** 功能实现库

这些库的层次关系如下图所示：




**OpenCore** 的各个库之间具有如下的关系：

**libopencorecommon.so** 是所有的库的依赖库，提供了公共的功能；

**libopencoreplayer.so** 和 **libopencoreauthor.so** 是两个并立的库，分别用于回放和记录，而且这两个库是 **OpenCore** 对外的接口库；

**libopencorenet_support.so** 提供网络支持的功能；

一些功能以插件（**Plug-In**）的方式放入 **Player** 中使用，每个功能使用两个库，一个实现具体功能，一个用于注册。

### libopencorecommon.so 库的结构

libopencorecommon.so 是整个 OpenCore 的核心库，它的编译控制的文件的路径为 pvcommon/Android.mk，这个文件使用递归的方式寻找子文件：

```
include $(BUILD_SHARED_LIBRARY)
include $(PV_TOP)//oscl/oscl/osclbase/Android.mk
include $(PV_TOP)//oscl/oscl/osclerror/Android.mk
include $(PV_TOP)//oscl/oscl/osclmemory/Android.mk
include $(PV_TOP)//oscl/oscl/osclutil/Android.mk
include $(PV_TOP)//oscl/pvlogger/Android.mk
include $(PV_TOP)//oscl/oscl/osclproc/Android.mk
include $(PV_TOP)//oscl/oscl/osclio/Android.mk
include $(PV_TOP)//oscl/oscl/osclregcli/Android.mk
include $(PV_TOP)//oscl/oscl/osclregserv/Android.mk
include $(PV_TOP)//oscl/unit_test/Android.mk
include $(PV_TOP)//oscl/oscl/osclib/Android.mk
include $(PV_TOP)//pvmi/pvmf/Android.mk
include $(PV_TOP)//baselibs/pv_mime_utils/Android.mk
include $(PV_TOP)//nodes/pvfileoutputnode/Android.mk
include $(PV_TOP)//baselibs/media_data_structures/Android.mk
include $(PV_TOP)//baselibs/threadsafe_callback_ao/Android.mk
include $(PV_TOP)//codecs_v2/utilities/colorconvert/Android.mk
include $(PV_TOP)//codecs_v2/audio/gsm_amr/amr_nb/common/Android.mk
include $(PV_TOP)//codecs_v2/video/avc_h264/common/Android.mk
```

这些被包含的Android.mk文件真正指定需要编译的文件，这些文件在Android.mk的目录及其子目录中。事实上，在 libopencorecommon.so 库中包含了以下内容：

- OSCL 的所有内容
- Pvmf 框架部分的内容（pvmi/pvmf/Android.mk）
- 基础库中的一些内容（baselibs）
- 编解码的一些内容
- 文件输出的 node（nodes/pvfileoutputnode/Android.mk）

从库的结构中可以看出，最终生成库的结构与 OpenCore 的层次关系并非完全重合。libopencorecommon.so 库中就包含了底层的 OSCL 的内容、PVMF 的框架以及 Node 和编解码的工具。

### libopencoreplayer.so 库的结构

libopencoreplayer.so 是用于播放的功能库，它的编译控制的文件的路径为 pvplayer/Android.mk，它包含了以下的内容：

```
include $(BUILD_SHARED_LIBRARY)
include $(PV_TOP)//engines/player/Android.mk
include $(PV_TOP)//codecs_v2/audio/aac/dec/util/getactualaacconfig/Android.mk
include $(PV_TOP)//codecs_v2/video/avc_h264/dec/Android.mk
include $(PV_TOP)//codecs_v2/audio/aac/dec/Android.mk
include $(PV_TOP)//codecs_v2/audio/gsm_amr/amr_nb/dec/Android.mk
include $(PV_TOP)//codecs_v2/audio/gsm_amr/amr_wb/dec/Android.mk
include $(PV_TOP)//codecs_v2/audio/gsm_amr/common/dec/Android.mk
include $(PV_TOP)//codecs_v2/audio/mp3/dec/Android.mk
include $(PV_TOP)//codecs_v2/utilities/m4v_config_parser/Android.mk
include $(PV_TOP)//codecs_v2/utilities/pv_video_config_parser/Android.mk
include $(PV_TOP)//codecs_v2/omx/omx_common/Android.mk
include $(PV_TOP)//codecs_v2/omx/omx_queue/Android.mk
```

include $(PV_TOP)//codecs_v2/omx/omx_h264/Android.mk
include $(PV_TOP)//codecs_v2/omx/omx_aac/Android.mk
include $(PV_TOP)//codecs_v2/omx/omx_amr/Android.mk
include $(PV_TOP)//codecs_v2/omx/omx_mp3/Android.mk
include $(PV_TOP)//codecs_v2/omx/factories/omx_m4v_factory/Android.mk
include $(PV_TOP)//codecs_v2/omx/omx_proxy/Android.mk
include $(PV_TOP)//nodes/common/Android.mk
include $(PV_TOP)//pvmi/content_policy_manager/Android.mk
include $(PV_TOP)//pvmi/content_policy_manager/plugins/oma1/passthru/Android.mk
include $(PV_TOP)//pvmi/content_policy_manager/plugins/common/Android.mk
include $(PV_TOP)//pvmi/media_io/pvmiofileoutput/Android.mk
include $(PV_TOP)//fileformats/common/parser/Android.mk
include $(PV_TOP)//fileformats/id3parcom/Android.mk
include $(PV_TOP)//fileformats/rawgsmamr/parser/Android.mk
include $(PV_TOP)//fileformats/mp3/parser/Android.mk
include $(PV_TOP)//fileformats/mp4/parser/Android.mk
include $(PV_TOP)//fileformats/rawaac/parser/Android.mk
include $(PV_TOP)//fileformats/wav/parser/Android.mk
include $(PV_TOP)//nodes/pvaacffparsernode/Android.mk
include $(PV_TOP)//nodes/pvmp3ffparsernode/Android.mk
include $(PV_TOP)//nodes/pvamrffparsernode/Android.mk
include $(PV_TOP)//nodes/pvmediaoutputnode/Android.mk
include $(PV_TOP)//nodes/pvomxvideodecnode/Android.mk
include $(PV_TOP)//nodes/pvomxaudiodecnode/Android.mk
include $(PV_TOP)//nodes/pvwavffparsernode/Android.mk
include $(PV_TOP)//pvmi/recognizer/Android.mk
include $(PV_TOP)//pvmi/recognizer/plugins/pvamrffrecognizer/Android.mk
include $(PV_TOP)//pvmi/recognizer/plugins/pvmp3ffrecognizer/Android.mk
include $(PV_TOP)//pvmi/recognizer/plugins/pvwavffrecognizer/Android.mk
include $(PV_TOP)//engines/common/Android.mk
include $(PV_TOP)//engines/adapters/player/framemetadatautility/Android.mk
include $(PV_TOP)//protocols/rtp_payload_parser/util/Android.mk
include $(PV_TOP)//android/Android.mk
include $(PV_TOP)//android/drm/oma1/Android.mk
include $(PV_TOP)//tools_v2/build/modules/linux_rtsp/core/Android.mk
include $(PV_TOP)//tools_v2/build/modules/linux_rtsp/node_registry/Android.mk
include $(PV_TOP)//tools_v2/build/modules/linux_net_support/core/Android.mk
include $(PV_TOP)//tools_v2/build/modules/linux_download/core/Android.mk
include $(PV_TOP)//tools_v2/build/modules/linux_download/node_registry/Android.mk
include $(PV_TOP)//tools_v2/build/modules/linux_mp4/core/Android.mk
include $(PV_TOP)//tools_v2/build/modules/linux_mp4/node_registry/Android.mk

**libopencoreplayer.so** 中包含了以下内容：

一些解码工具
文件的解析器（**mp4**）
解码工具对应的 **Node**
**player** 的引擎部分（**engines/player/Android.mk**）
为 **Android** 的 **player** 适配器（**android/Android.mk**）
识别工具（**pvmi/recognizer**）
编解码工具中的 **OpenMax** 部分**(codecs_v2/omx)**

对应几个插件 Node 的注册

libopencoreplayer.so 中的内容较多，其中主要为各个文件解析器和解码器，PVPlayer 的核心功能在 engines/player/Android.mk 当中，而 android/Android.mk 的内容比较特殊，它是在 PVPlayer 之上构建的一个为 Android 使用的播放器。

### libopencoreauthor.so 库的结构

libopencoreauthor.so 是用于媒体流记录的功能库，它的编译控制的文件的路径为 pvauthor/Android.mk，它包含了以下的内容：

```
include $(BUILD_SHARED_LIBRARY)
include $(PV_TOP)//engines/author/Android.mk
include $(PV_TOP)//codecs_v2/video/m4v_h263/enc/Android.mk
include $(PV_TOP)//codecs_v2/audio/gsm_amr/amr_nb/enc/Android.mk
include $(PV_TOP)//codecs_v2/video/avc_h264/enc/Android.mk
include $(PV_TOP)//fileformats/mp4/composer/Android.mk
include $(PV_TOP)//nodes/pvamrencnode/Android.mk
include $(PV_TOP)//nodes/pvmp4ffcomposernode/Android.mk
include $(PV_TOP)//nodes/pvvideoencnode/Android.mk
include $(PV_TOP)//nodes/pvavcencnode/Android.mk
include $(PV_TOP)//nodes/pvmediainputnode/Android.mk
include $(PV_TOP)//android/author/Android.mk
```

libopencoreauthor.so 中包含了以下内容：

一些编码工具（视频流 H263、H264，音频流 Amr）

文件的组成器（mp4）

编码工具对应的 Node

表示媒体输入的 Node（nodes/pvmediainputnode/Android.m）

author 的引擎部分（engines/author/Android.mk）

为 Android 的 author 适配器（android/author/Android.mk）

libopencoreauthor.so 中主要为各个文件编码器和文件组成器，PVAuthor 的核心功能在 engines/author/Android.mk 当中，而 android/author/Android.mk 是在 PVAuthor 之上构建的一个为 Android 使用的媒体记录器。

### 其他库

另外的几个库的 Android.mk 文件的路径如下所示：

网络支持库 libopencorenet_support.so：

tools_v2/build/modules/linux_net_support/core/Android.mk

MP4 功能实现库 libopencoremp4.so 和注册库 libopencoremp4reg.so：

tools_v2/build/modules/linux_mp4/core/Android.mk

tools_v2/build/modules/linux_mp4/node_registry/Android.mk

RTSP 功能实现库 libopencorertsp.so 和注册库 libopencorertspreg.so：

tools_v2/build/modules/linux_rtsp/core/Android.mk

tools_v2/build/modules/linux_rtsp/node_registry/Android.mk

下载功能实现库 libopencoredownload.so 和注册库 libopencoredownloadreg.so：

tools_v2/build/modules/linux_download/core/Android.mk

tools_v2/build/modules/linux_download/node_registry/Android.mk

# 第三部分 OpenCore OSCL 简介

OSCL，全称为 Operating System Compatibility Library (操作系统兼容库)，它包含了一些在不同操作系统中移植层的功能，其代码结构如下所示：

```
oscl/oscl
|-- config          ：配置的宏
|-- makefile
|-- makefile.pv
|-- osclbase        ：包含基本类型、宏以及一些 STL 容器类似的功能
|-- osclerror       ：错误处理的功能
|-- osclio          ：文件 IO 和 Socket 等功能
|-- osclib          ：动态库接口等功能
|-- osclmemory      ：内存管理、自动指针等功能
|-- osclproc        ：线程、多任务通讯等功能
|-- osclregcli      ：注册客户端的功能
|-- osclregserv     ：注册服务器的功能
`-- osclutil        ：字符串等基本功能
```

在 oscl 的目录中，一般每一个目录表示一个模块。OSCL 对应的功能是非常细致的，几乎对 C 语言中每一个细节的功能都进行封装，并使用了 C++的接口提供给上层使用。事实上，OperCore 中的 PVMF、Engine 部分都在使用 OSCL，而整个 OperCore 的调用者也需要使用 OSCL。

在 OSCL 的实现中，很多典型的 C 语言函数被进行了简单的封装，例如：osclutil 中与数学相关的功能在 oscl_math.inl 中被定义成为了内嵌（inline）的函数：

```
OSCL_COND_EXPORT_REF OSCL_INLINE double oscl_log(double value)
{
        return (double) log(value);
}
OSCL_COND_EXPORT_REF OSCL_INLINE double oscl_log10(double value)
{
        return (double) log10(value);
}
OSCL_COND_EXPORT_REF OSCL_INLINE double oscl_sqrt(double value)
{
        return (double) sqrt(value);
}
```

oscl_math.inl 文件又被 oscl_math.h 所包含，因此其结果是 oscl_log()等功能的使用等价于原始的 log()等函数。

很多 C 语言标准库的句柄都被定义成为了 C++类的形式，实现由一些繁琐，但是复杂性都不是很高。以 osclib 为例，其代码结构如下所示：

```
oscl/oscl/osclib/
|-- Android.mk
|-- build
|   `-- make
|       `-- makefile
`-- src
    |-- oscl_library_common.h
    |-- oscl_library_list.cpp
    |-- oscl_library_list.h
    |-- oscl_shared_lib_interface.h
```

```
    |-- oscl_shared_library.cpp
    `-- oscl_shared_library.h
```

oscl_shared_library.h 是提供给上层使用的动态库的接口功能，它定义的接口如下所示：

```
class OsclSharedLibrary
{
public:
        OSCL_IMPORT_REF OsclSharedLibrary();
        OSCL_IMPORT_REF OsclSharedLibrary(const OSCL_String& aPath);
        OSCL_IMPORT_REF ~OsclSharedLibrary();
        OSCL_IMPORT_REF OsclLibStatus LoadLib(const OSCL_String& aPath);
        OSCL_IMPORT_REF OsclLibStatus LoadLib();
        OSCL_IMPORT_REF void SetLibPath(const OSCL_String& aPath);
                OSCL_IMPORT_REF OsclLibStatus QueryInterface(const OsclUuid& aInterfaceId,
OsclAny*& aInterfacePtr);
        OSCL_IMPORT_REF OsclLibStatus Close();
        OSCL_IMPORT_REF void AddRef();
        OSCL_IMPORT_REF void RemoveRef();
}
```

这些接口显然都是与库的加载有关系的，而在oscl_shared_library.cpp 中其具体的功能是使用dlopen()等函数来实现的。

# 第四部分 文件格式处理和编解码部分简介

在多媒体方面，文件格式的处理和编解码（Codec）是很基础的两个方面的内容。多媒体应用的两个方面是媒体的播放（PlayBack）和媒体的记录（Recording）。

在媒体的播放过程中，通常情况是从对媒体文件的播放，必要的两个步骤为文件的解析和媒体流的解码。例如对于一个 mp4 的文件，其中可能包括 AMR 和 AAC 的音频流，H263、MPEG4 以及 AVC（H264）的视频流，这些流被封装在 3GP 的包当中，媒体播放器要做的就是从文件中将这些流解析出来，然后对媒体流进行解码，解码后的数据才可以播放。

在媒体的记录过程中，通过涉及到视频、音频、图像的捕获功能。对于将视频加音频录制成文件功能，其过程与播放刚好相反，首先从硬件设备得到视频和音频的媒体流，然后对其进行编码，编码号的流还需要被分层次写入到文件之中，最终得到组成好的文件。

OpenCore 有关文件格式处理和编解码部分两部分的内容，分别在目录 fileformats 和 codecs_v2 当中。这两部分都属于基础性的功能，不涉及具体的逻辑，因此它们被别的模块调用来使用，例如：构建各种 Node。

## 4.1 文件格式的处理

由于同时涉及播放文件和记录文件两种功能，因此 OpenCore 中的文件格式处理有两种类型，一种是 parser（解析器），另一种是 composer（组成器）。

fileformats 的目录结构如下所示：

```
fileformats
|-- avi
|   `-- parser
|-- common
|   `-- parser
|-- id3parcom
|   |-- Android.mk
|   |-- build
```

```
|    |-- include
|    `-- src
|-- mp3
|    `-- parser
|-- mp4
|    |-- composer
|    `-- parser
|-- rawaac
|    `-- parser
|-- rawgsmamr
|    `-- parser
`-- wav
     `-- parser
```

目录包含各个子目录中，它们对应的是不同的文件格式，例如 **mp3**、**mp4** 和 **wav** 等。

## 4.2 编解码

编解码部分主要针对 **Audio** 和 **Video**，**codecs_v2** 的目录结构如下所示：

```
codecs_v2
|-- audio
|    |-- aac
|    |-- gsm_amr
|    |-- mp3
|    `-- sbc
|-- omx
|    |-- factories
|    |-- omx_aac
|    |-- omx_amr
|    |-- omx_common
|    |-- omx_h264
|    |-- omx_m4v
|    |-- omx_mp3
|    |-- omx_proxy
|    `-- omx_queue
|-- utilities
|    |-- colorconvert
|    |-- m4v_config_parser
|    `-- pv_video_config_parser
`-- video
     |-- avc_h264
     `-- m4v_h263
```

在 **audio** 和 **video** 目录中，对应了针对各种流的子目录，其中可能包含 **dec** 和 **enc** 两个目录，分别对应解码和编码。**video** 目录展开后的内容如下所示：

```
`-- video
     |-- avc_h264
     |    |-- common
     |    |-- dec
     |    |-- enc
     |    `-- patent_disclaimer.txt
     `-- m4v_h263
```

```
|-- dec
|-- enc
`-- patent_disclaimer.txt
```

codecs_v2 目录的子目录 omx 实现了一个 khronos

OpenMAX 的功能。OpenMAX 是一个多媒体应用程序的框架标准，由 NVIDIA 公司和 Khronos 在 2006 年推出。OpenMAX IL 1.0（集成层）技术规格定义了媒体组件接口，以便在嵌入式器件的流媒体框架中快速集成加速式编解码器。

OpenMAX 的设计实现可以让具有硬件编辑码功能的平台提供统一的接口和框架，在 OpenMAX 中可以直接使用硬件加速的进行编解码乃至输出的功能，对外保持统一的接口。但是在此处的 OpenMAX 则是一个纯软件的实现。