**总结三**

1. **项目里方法**
2. 监听设备连接

首先通过IMBDeviceConnection单例方法获取一个单例对象

之后通过其对象方法- startListen进行监听

在初始化方法中添加监听相应的通知，分别是:

DeviceConnectedNotification（设备连接成功通知）

DeviceDisConnectedNotification（设备断开连接通知）

DeviceChangeNotification（连接设备改变通知）

DeviceNeedPasswordNotification（连接的设备需要密码通知）

DeviceIpodLoadCompleteNotification（连接设备数据加载完成后通知）

1. 设备连接成功后获取设备内Itunes数据

通过DeviceIpodLoadCompleteNotification通知传递过来的userInfo数据获取IMBBaseInfo对象，之后通过此对象获取对应的IMBiPod对象，之后通过iPod对象获取对应的IMBInformation对象，最后通过information对象的对象方法’- getTrackArrayByMediaTypes:’获取到设备Itunes数据信息

- (void)deviceIpodLoadComplete:(NSNotification \*)notification {

NSDictionary \*userInfo = [notification userInfo];

IMBBaseInfo \*baseInfo = userInfo[@"DeviceInfo"];

self.sizeLabel.stringValue = [NSString stringWithFormat:@"%@: %.01f GB Free/%.01f GB Total",baseInfo.deviceName,baseInfo.kyDeviceSize/1024.0f/1024.0f/1024.0f,baseInfo.allDeviceSize/1024.0f/1024.0f/1024.0f];

IMBiPod \*ipod = [[IMBDeviceConnection singleton] getIPodByKey:baseInfo.uniqueKey];

if (ipod) {

\_information = [[IMBInformation alloc] initWithiPod:ipod];

\_dataArray = [[NSMutableArray alloc] initWithArray:[\_information getTrackArrayByMediaTypes:[IMBCommonEnum categoryNodeToMediaTyps:Category\_Music]]];

[\_tableView reloadData];

}

1. 显示获取到的数据信息

通过tableView将获取到的设备Itunes数据信息显示到界面之上

1. **实际调用的底层C代码**
2. 监听设备连接

有关连接设备相关的API几乎都来自于MobileDevice.framework这个苹果系统自带的私有框架，由于此框架属于系统私有框架，所以不对外提供调用的接口，一切有关此框架的使用接口均来自于对苹果自带相应功能软件的反编译获得。

首先：对设备进行监听，用到的是MobileDevice.framework框架里的下面这个函数

mach\_error\_t AMDeviceNotificationSubscribe(

am\_device\_notification\_callback callback,

uint32\_t unused0,

uint32\_t unused1,

void \*callback\_data,

am\_device\_notification \*notification);

使用：

AMDeviceNotificationSubscribe(notify\_callback, 0, 0, self, &\_notification);

其中notify\_callback为自己写好的一个回调函数，用于响应设备连接的相关改变。传入的\_notification主要用于此后监听的取消

取消监听则使用框架内的下面这个函数

mach\_error\_t AMDeviceNotificationUnsubscribe(

am\_device\_notification subscription);

1. 设备连接成功获取设备信息

设备连接相对应的所有消息将由下面这个函数对其作出相应和接受相应的消息

static void notify\_callback(struct am\_device\_notification\_callback\_info \*info, void\* arg)

此函数中的info是一个此框架内的结构体

struct am\_device\_notification\_callback\_info {

am\_device dev; // 0 device

uint32\_t msg; // 4 one of adnci\_msg

} \_\_attribute\_\_ ((packed));

这个结构体包含了获取设备相关信息的am\_device参数和一个连接设备状态的msg消息参数

这里的am\_device是一个框架内自带的结构体句柄

typedef struct \_am\_device \*am\_device;

根据回调函数里得到的am\_device\_notification\_callback\_info句柄中的msg判断此刻连接设备的状态及相关的一系列信息

msg里对应的是一个int32类型的数值，起对应的数值是下面enum中的一个

typedef enum {

ADNCI\_MSG\_CONNECTED = 1,

ADNCI\_MSG\_DISCONNECTED = 2,

ADNCI\_MSG\_UNSUBSCRIBED = 3

} adnci\_msg;

1则为设备连接，2则为设备断开连接，3则为取消对设备的监听

设备连接成功后，可以通过回调参数info里的dev获取到一系列设备相关信息，主要通过过框架内的下面函数进行获取

CFStringRef AMDeviceCopyValue(am\_device device, CFStringRef domain,CFStringRef key);

其中的要获取设备内对应的设备信息就必须得通过对应的domain和key值来进行获取，其中domain和key都唯一对应着一条相应的设备信息。传入不同的device就将获取对应的设备上的数据信息

通过这个函数可以获取例如设备名、udid、产品名、磁盘容量和磁盘可用容量等信息。

1. 复制设备内ItunesCDB文件至项目内(非ios系统的名字为ItunesDB)

问题:NSFileManager也有一个对象方法，专门用于copy文件，为什么不用，而要自己写一个copy方法呢？

Copy方法主要用NSFileManager的write和read对象方法进行操作。

首先用下列框架内函数得到一个afc\_connection的变量

afc\_error\_t AFCConnectionOpen(am\_service handle,uint32\_t io\_timeout,afc\_connection \*conn);

在对文件进行操作之前，使用下面函数判断当下能否对文件进行操作，当返回值为0 的时候，则可以进行操作

afc\_error\_t AFCFileRefOpen(afc\_connection conn, const char \*path, uint64\_t mode,afc\_file\_ref \*ref);

之后在conn有值的情况下才能保证对文件进行相应的操作

copy的文件是位于设备内路径为/iTunes\_Control/iTunes/iTunesCDB(此路径为定死的路径)名为ItunesCDB文件。Copy到的本地路径为项目内的随意路径即可。复制成功后就进行下面解析文件数据的操作，操作结束之后，就将复制进本地项目内的ItunesCDB文件删除。

1. 解析ItunesCDB文件数据

这个确实比较难，也在网上百度了相关的知识，但是也是极少量的信息。Itunes主要是一个二进制文件，如果解析的话，就得一个节点一个节点的进行解析，并且解析的过程步骤也必须得按照文件内的节点和每个节点对应的字节长度进行解析，有一点的出错就将导致后续的解析失败。

iTunesCDB 是一个二进制的文件，其中是用节点来组织的，每个节点都包含特定的信息，比如节点类型，节点头大小，子节点个数等等。有点像xml，但是和xml 又不一样，xml 是闭合的。

MHBD 头

二进制文件的头四个字节mhbd 是一个标识，根据特定的标识不同的解析，在解析的时候只要关心自己需要的字段，其他不明白的字段读出来之后保存起来就可以了。下面介绍一下我在解析的时候关注的字段

1)   4字节的头标识  mhbd

2)   4字节的头长度

3)  4字节的总长度

如果是解析的话只需要关注这三个字段就可以了。

这个文件的解析难度还是挺大的，我这周搞了好几天，还是没法解析成功，应该也只能用项目里写好的文件，或者周末再继续试着解析解析。

可以新建一个对应的模型类，对解析出来的数据进行存储。项目里用的是IMBTrack进行存储。

1. 显示数据

显示数据同上。

**难点：**

1、文件copy时的AFCConnectionOpen和AFCFileRefOpen两个函数总是会返回错误的数据，具体原因不明，自己猜测可能是传入的某些参数有误。

2、文件copy时如果突然中断设备连接时，文件copy的相应操作

3、ItunesCDB文件解析：到现在没法解析成功，主要原因难度系数比较大，或者说是自己技术能力有限或者对这方面技术知识了解不够。

**问题：**

1、下面两个函数的有什么区别，什么时候用哪一个

mach\_error\_t AMDeviceStartService(am\_device device,CFStringRef service\_name,am\_service \*handle,uint32\_t \*unknown);

mach\_error\_t AMDeviceSecureStartService(am\_device device, CFStringRef service\_name, uint32\_t \*unknown, am\_service \*handle);

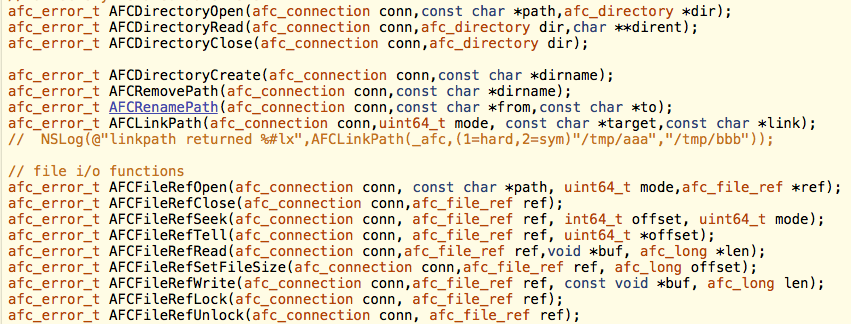
2、下面一个函数有什么作用，主要用在什么地方

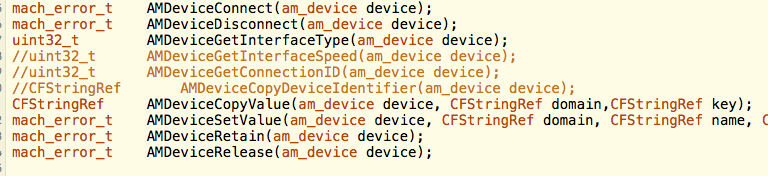
mach\_error\_t AMDeviceStartSession(am\_device device);

3、下面函数中的am\_service的对应参数应该传入什么样的值才会返回对应正确的值

afc\_error\_t AFCConnectionOpen(am\_service handle,uint32\_t io\_timeout,afc\_connection \*conn);

1. 、框架内以AFC开头的函数中的这个AFC是什么意思，还有以AM开头的函数的AM是什么意思，挺想搞明白的





5、下面两个函数的作用（按字面意思能理解为匹配设备是否是同一个，但是看使用的地方，感觉并不是这个意思）

int AMDevicePair(am\_device device);

int AMDeviceIsPaired(am\_device device);

**收获：**

1. 明白了很多功能的底层实现和MobilDevice里的部分API的功能作用和使用
2. 明白了公司项目的很多功能上的实现大部分都是借助于系统的某些私有框架，而这些私有框架则不提供对外接口，需要自行根据系统自带的具有相应功能的软件进行反编译从而获取框架内相应的接口API
3. 对搞明白这些框架的接口API产生了兴趣，挺想要弄清楚这些东西，也想有时间自己学学苹果方面反编译的知识
4. 算是重振了自己今年这么长一段时间不想学习的萎靡不振的心，激发了自己的学习激情