Amlogic mx 平台播放器的内存泄漏优化

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **变更（+/-）说明** | **作者** | **版本号** | **日期** | **批准** |
| 1 | 初稿 | 柳仙明 | V1.0 | 2016-3-21 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 问题背景

Amlogic mx平台的StreamingMediaPlayer在播放过程中出现过内存增长，但是退出播放或者kill掉StreamingMediaPlayer播放器，内存开始减少但是并未回到初始内存的值，并且隔了一个不小的内存值。因此，我们怀疑播放器内存出现泄漏。由于应用层开始时已经排查，于是我开始查询Framework层以及amlogic层的源码，一步步来找出错误的根源。

# 问题原因

首先，我上网查询了一些资料，了解了一下内存泄漏的概念以及原因。安卓Framework 层和库层主要是c/c++编写的，因此我们考虑c/c++内存泄漏的原因。

## 内存泄漏的简介

指由于疏忽或错误造成程序未能释放已经不再使用的内存的情况。内存泄漏并非指内存在物理上的消失，而是[应用程序](http://baike.baidu.com/view/330120.htm)分配某段内存后，由于设计错误，失去了对该段内存的控制，因而造成了内存的浪费。

## C/c++内存泄漏的主要原因

1. 堆内存泄漏（Heap leak）。堆内存指的是程序运行中根据需要分配通过malloc,realloc,new等从堆中分配的一块内存，再是完成后必须通过调用对应的 free或者delete 删掉。如果程序的设计的错误导致这部分内存没有被释放，那么此后这块内存将不会被使用，就会产生Heap Leak
2. 系统资源泄露（Resource Leak）.主要指程序使用系统分配的资源比如 Bitmap,handle ,socket等没有使用相应的函数释放掉，导致系统资源的浪费，严重可导致系统效能降低，系统运行不稳定。

# 查找方法

明确内存泄漏产生的原因之后，我就开始根据Amlogic给出的调试方案开始一步步的找出错误代码根源。

## 排查方案步骤

1. 烧录Debug版本的软件并安装进机顶盒
2. 打开secureCRT连接导师的服务器进入项目，并配置好环境变量、lunch g18ref-eng
3. 开机在机顶盒上setprop libc.debug.malloc 1然后再killall mediaserver
4. Top/grep mediaserver查找出进程号，cat /proc/$(pidof mediaserver)/maps >(pid)maps，并且将进程对应的maps转存在data/tmp目录下，以便查看以及机顶盒重启时丢失。
5. 程序运行前 dumpsys media.player -m > filename1存到data/tmp目录下
6. 进入apk测试一段时间
7. 出apk或者手动kill apk 进程
8. dumpsys media.player -m > filename2 存到data/tmp目录下
9. 导出maps filename1 filename2三个文件到电脑，分析比较得出结果。

## 比较方法

这里filename1和filename2为这两个状态内存申请情况，比较差异，多出来的就是泄漏的了

比如

size 702, dup 13 0x40153fc8, 0x4012aa80, 0x4012b91c, 0x4012b95c, 0x400ac072, 0x41ce2ba6, 0x4013092a, 0x40066952, 0x400707c8,0x40208268, 0x40207dce, 0x401554d4, 0x40154b74

这里0x40153fc8, 0x4012aa80, 0x4012b91c, 0x4012b95c, 0x400ac072, 0x41ce2ba6, 0x4013092a, 0x40066952, 0x400707c8,0x40208268, 0x40207dce, 0x401554d4, 0x40154b74

为调用的backtrace

从最后一个开始，地址0x40154b74，从之前的maps里面找出这个值落在哪个库的map段

40147000-4018c000 r-xp 00000000 fa:10 41053 /system/lib/libc.so找出offset 0x40154b74- 40147000= DB74 查出对应代码行

cuihuijuan@ubuntu-sdmc:~/mx\_fw\_0428\_new$ addr2line –e out/target/product/g18ref/symbols/system/lib/libc.so db74

/home/cuihuijuan/mx\_fw\_0428\_new/bionic/libc/bionic/pthread.c:348然后就查看pthread.c的348行，在附近找寻错误。以此类推，往前一步步开始找寻错误。

# 查找结果

## 见附录。通过分析查找的结果，可以看出播放器的调用过程大致如下:



## 错误发现

查找内存泄漏代码时，发现了源码中的一处错误:

0x41ce2ba6

41ce1000-41ce4000 r-xp 00000000 fa:10 40985 /system/lib/hw/audio.usb.amlogic.so 1ba6

/home/cuihuijuan/mx\_fw\_0428\_new/hardware/amlogic/audio/usb\_audio\_hw.c:294

代码如下:

if (out->out\_config.rate != pcm\_out\_config.rate){

err = create\_resampler(DEFAULT\_OUT\_SAMPLING\_RATE,

out->out\_config.rate,

out->out\_config.channels,

RESAMPLER\_QUALITY\_DEFAULT,

NULL,

&out->resampler);

if (err != 0)

return -ENOMEM;

out->buffer = malloc(RESAMPLER\_BUFFER\_SIZE); /\* todo: allow for reallocing \*/

if (!out->buffer)

return -ENOMEM;

}

out->out\_pcm = pcm\_open(adev->card, adev->card\_device, PCM\_OUT,&out->out\_config);

if (!pcm\_is\_ready(out->out\_pcm)) {

ALOGE("pcm\_open() failed: %s", pcm\_get\_error(out->out\_pcm));

pcm\_close(out->out\_pcm);

adev->active\_output = NULL;

if(out->buffer)

{

free(out->buffer);//没有释放掉 ，将其释放

out->buffer=NULL;

}

return -ENOMEM;

}

return 0;

}

# 总结

其实内存泄漏的原因可以概括为：调用了malloc/new等内存申请的操作，但缺少了对应的free/delete，总之就是，malloc/new比free/delete的数量多。我们在编程时需要注意这点。要避免内存泄漏可以总结为以下几点：

* 程序员要养成良好习惯，保证malloc/new和free/delete匹配；
* 在使用内存之前检查指针是否为NULL。如果指针p 是函数的参数，那么在函数的入口处用assert(p!=NULL)进行检查。如果是用malloc 或new 来申请内存，应该用if(p==NULL)或if(p!=NULL)进行防错处理；
* 使用free 或delete 释放了内存后，应将指针设置为NULL。否则产生“野指针”。