MKV文件被扫描成Audio文件的案例分析

Last edited by **caoquanli** 1 month ago

MKV格式文件调查

本编文章主要以在Android平台下，一个MKV格式的视频文件出现在audio数据库中这个情况展开讲述视频文件的扫描。

首先媒体文件的扫描在上面已经叙述过，这里不再讲这个，这里说一下是音视频文件的MimeType，在扫描初期，这个type是根据文件的后缀名来取的，所有文件格式的说明都在MediaFile.java中有定义。

- MediaFile.java 这个文件的具体内容大家可以取看看你，里面是所有文件的type标识，以及在定义这个文件的时候都会set好这个文件的Mimetype。

测试文件太大这里就不放了。

所以呢，在扫描初期，他的MimeType都是在这里获得，从这里拿到的type肯定不会出错，现在问题呢就出现在这里了，他为什么会出现在audio数据库，往下追踪。

每个文件的MimeType在处理文件时也会在重新set一次，在他获取到的数据表里，现在呢，我们就取看看他是怎么获取文件的mimetype，顺带着了解一下视频文件的解码。

经过stageFright框架，最终在MediaExtractor.cpp中创建解析器，解析文件，在这里他解析器的创建也是根据在java层获取到的MimeType，对于MKV文件

```
 MediaExtractor *ret = NULL;
if (!strcasecmp(mime, MEDIA_MIMETYPE_CONTAINER_MPEG4)
        || !strcasecmp(mime, "audio/mp4")) {
    ret = new MPEG4Extractor(source);
} else if (!strcasecmp(mime, MEDIA_MIMETYPE_AUDIO_MPEG)) {
    ret = new MP3Extractor(source, meta);
} else if (!strcasecmp(mime, MEDIA_MIMETYPE_AUDIO_AMR_NB)
        || !strcasecmp(mime, MEDIA_MIMETYPE_AUDIO_AMR_WB)) {
    ret = new AMRExtractor(source);
} else if (!strcasecmp(mime, MEDIA_MIMETYPE_AUDIO_FLAC)) {
    ret = new FLACExtractor(source);
} else if (!strcasecmp(mime, MEDIA_MIMETYPE_CONTAINER_WAV)) {
    ret = new WAVExtractor(source);
} else if (!strcasecmp(mime, MEDIA_MIMETYPE_CONTAINER_OGG)) {
    ret = new OggExtractor(source);
} else if (!strcasecmp(mime, MEDIA_MIMETYPE_CONTAINER_MATROSKA)) {
    ret = new MatroskaExtractor(source);
} else if (!strcasecmp(mime, MEDIA_MIMETYPE_CONTAINER_MPEG2TS)) {
    ret = new MPEG2TSExtractor(source);
} else if (!strcasecmp(mime, MEDIA_MIMETYPE_AUDIO_AAC_ADTS)) {
    ret = new AACExtractor(source, meta);
} else if (!strcasecmp(mime, MEDIA_MIMETYPE_CONTAINER_MPEG2PS)) {
    ret = new MPEG2PSExtractor(source);
} else if (!strcasecmp(mime, MEDIA_MIMETYPE_AUDIO_MIDI)) {
    ret = new MidiExtractor(source);
}
```

上面就是创建具体解析器的方法，对于MKV格式的会创建MatroskaExtractor解析器，进入这个解析器，看看他到底是怎么搞得。

首先呢，对于音频文件，大家都知道是有音轨得，现在得技术，都是把视频track和音频track分开得，所以呢，一个是视频至少是有两条track得，这里面有个addTracks这个函数，其实，这个解析器只是一个信息收集器，并不是真正得解析器，具体解析，那需要取看底层源码得mkvparser，现在这里就是收集一些数据，进行分析，在这个函数中，他会对每一条track拿出来看，是否支持解码，每一条track都是胡对应不同得codec的，现在对于我的那个文件

他的两条track的codecID分别是MS/VFW/FOURCC和A_AAC，前者是视频track后者是音频的track，现在呢问题就出在这个视频track上了。具体的获得方式请学习一下视频编解码以及用UE查看。现在看一下处理视频track的代码

```
case VIDEO_TRACK:
           {
               const mkvparser::VideoTrack *vtrack =
                   static_cast<const mkvparser::VideoTrack *>(track);

               if (!strcmp("V_MPEG4/ISO/AVC", codecID)) {
```

```
            meta->setCString(kKeyMIMEType, MEDIA_MIMETYPE_VIDEO_AVC);
            meta->setData(kKeyAVCC, 0, codecPrivate, codecPrivateSize);
        } else if (!strcmp("V_MPEG4/ISO/ASP", codecID)) {
            if (codecPrivateSize > 0) {
                meta->setCString(
                        kKeyMIMEType, MEDIA_MIMETYPE_VIDEO_MPEG4);
                addESDSFromCodecPrivate(
                        meta, false, codecPrivate, codecPrivateSize);
            } else {
                ALOGW("%s is detected, but does not have configuration.",
                        codecID);
                continue;
            }
        } else if (!strcmp("V_VP8", codecID)) {
            meta->setCString(kKeyMIMEType, MEDIA_MIMETYPE_VIDEO_VP8);
        } else if (!strcmp("V_VP9", codecID)) {
            meta->setCString(kKeyMIMEType, MEDIA_MIMETYPE_VIDEO_VP9);
            if (codecPrivateSize > 0) {
              // 'csd-0' for VP9 is the Blob of Codec Private data as
              // specified in http://www.webmproject.org/vp9/profiles/.
              meta->setData(
                      kKeyVp9CodecPrivate, 0, codecPrivate,
                      codecPrivateSize);
            }
        } else {
            ALOGW("%s is not supported.", codecID);
            continue;
        }

        const long long width = vtrack->GetWidth();
        const long long height = vtrack->GetHeight();
        if (width <= 0 || width > INT32_MAX) {
            ALOGW("track width exceeds int32_t, %lld", width);
            continue;
        }
        if (height <= 0 || height > INT32_MAX) {
            ALOGW("track height exceeds int32_t, %lld", height);
            continue;
        }
        meta->setInt32(kKeyWidth, (int32_t)width);
        meta->setInt32(kKeyHeight, (int32_t)height);

        // setting display width/height is optional
        const long long displayUnit = vtrack->GetDisplayUnit();
        const long long displayWidth = vtrack->GetDisplayWidth();
        const long long displayHeight = vtrack->GetDisplayHeight();
        if (displayWidth > 0 && displayWidth <= INT32_MAX
                && displayHeight > 0 && displayHeight <= INT32_MAX) {
            switch (displayUnit) {
            case 0: // pixels
                meta->setInt32(kKeyDisplayWidth, (int32_t)displayWidth);
                meta->setInt32(kKeyDisplayHeight, (int32_t)displayHeight);
                break;
            case 1: // centimeters
            case 2: // inches
            case 3: // aspect ratio
            {
                // Physical layout size is treated the same as aspect ratio.
                // Note: displayWidth and displayHeight are never zero as they are
                // checked in the if above.
                const long long computedWidth =
                        std::max(width, height * displayWidth / displayHeight);
                const long long computedHeight =
                        std::max(height, width * displayHeight / displayWidth);
                if (computedWidth <= INT32_MAX && computedHeight <= INT32_MAX) {
                    meta->setInt32(kKeyDisplayWidth, (int32_t)computedWidth);
                    meta->setInt32(kKeyDisplayHeight, (int32_t)computedHeight);
                }
                break;
            }
            default: // unknown display units, perhaps future version of spec.
                break;
            }
        }
```

```
                getColorInformation(vtrack, meta);

                break;
            }
```

这就出问题了，他这里会有个mTracks数组，这里对上面的track他是没有解码方式去解码的，所以这里的mTrack size就变成了1，对于出现这样的情况，在返回的时候StagefrightMetadataRetriever中会处理，他有个parseMetaData()函数，这个函数会具体判断它的track数，然后如果是1.看看代码

```
    if (numTracks == 1) {
        const char *fileMIME;
        if (meta->findCString(kKeyMIMEType, &fileMIME) &&
                !strcasecmp(fileMIME, "video/x-matroska"))               {
            sp<MetaData> trackMeta = mExtractor-   >getTrackMetaData(0);
            const char *trackMIME;
            CHECK(trackMeta->findCString(kKeyMIMEType, &trackMIME));

            if (!strncasecmp("audio/", trackMIME, 6)) {
                // The matroska file only contains a single audio track,
                // rewrite its mime type.
                mMetaData.add(
                        METADATA_KEY_MIMETYPE, String8("audio/x-matroska"));
            }
        }
    }
```

相信大家也能看得懂这上面得代码，从之前得到得track，只能匹配到audioTrack的codecID，所以他会把MetaData中的METADATA_KEY_MIMETYPE，强制设置为"audio/x-matroska"，这里就变了，这个MimeType就让他变成audio了，在最后set过去得MimeType正是这里获取到得type，因此到这里大家也就明白了为什么这个文件会扫出问题，归根到底还是本地解码器不支持解码。