AndroidManifest.xml 二进制文件格式分析

MindMac

2014/11/9

AndroidManifest.xml 文件的二进制格式没有公开文档，以下是根据 AXMLPrinter2 的源码进行的分析，可能会存在错误！

最终的 AndroidManifest.xml 文件二进制格式如 图 1 所示。

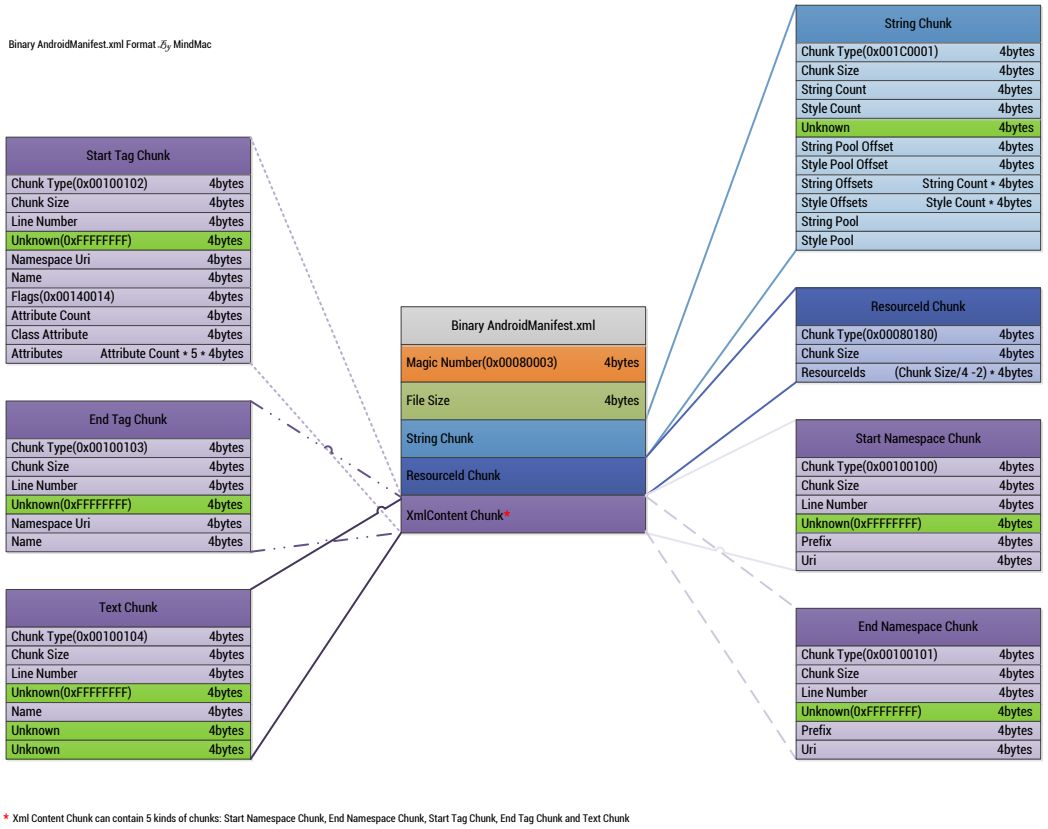

图 1 AndroidManifest.xml 二进制文件格式

下面根据一个实际的 AndroidManifest 文件进行对应的分析，编码前的文件内容如图 2

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.test"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="19" />

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.android.test.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

图 2 编码前的 AndroidManifest.xml 文件内容

所示，注意行号值，行号信息会反映在二进制文件内容中。

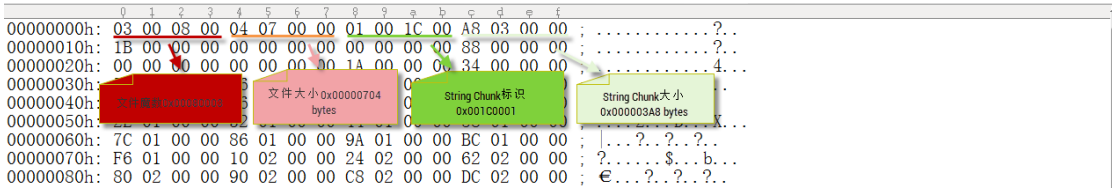

图 3 AndroidManifest 二进制内容

根据 图 1 中的文件格式对照图 3 可以发现二进制 XML 文件魔数为 0x00080003，该文件大小为 0x00000704，即 1796 bytes；从 0x00000008 开始为 String Chunk 的内容，首先是 String Chunk 的标识，值为 0x001C0001，String Chunk 的大小为 0x000003A8，即 936 bytes，所以这部分内容区间为 0x00000008-0x000003AF。0x00000010 开始的 4 个字节内容为字符串的个数，因此本文件中字符个数为 0x0000001B，即一共有 27 个字符串，这些字符串存放在相对于 String Chunk 开头偏移量为 0x00000088(0x0000001C 处的 4 个字节内容)的地方，即 0x00000008 + 0x0000000088 = 0x00000090。0x00000014 开始处的 4 个字节内容为 Style 个数，至于 Style 为何物，因为找了很多 APK，发现该处的值始终为 0x00000000，也就是没有 Style 内容，所以暂时无法确定 Style 具体意思。0x00000018 开始处的 4 个字节内容始终为 0x00000000，具体含义不确定。0x00000020 开始处的 4 字节内容为 Style 内容相对于 String Chunk 开头位置的偏移量，因为 Style 个数为 0，所以此处的值为 0x00000000。从 0x00000024 位置开始，后面连续跟着 27 个字符串的偏移量（相对于字符串内容的起始位置，在本次分析的 AndroidManifest 中即为 0x00000090），第一个字符串偏移量当然为 0 了，第二个字符串开始位置为 0x00000090+0x0000001A=0x000000AA。

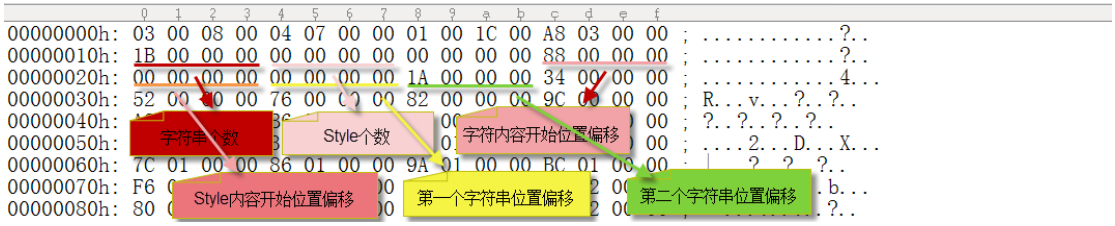

图 4 AndroidManifest 字符串以及 Style

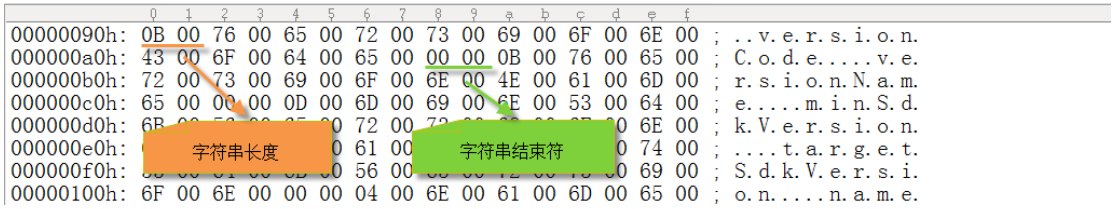

图 5 字符串内容格式

图 5 为 AndroidManifest 中字符串的内容格式，开始 2 个字节内容为字符串长度，该长度指的是字符串中字符个数，而非字节数，因此此处字符串字符个数为 0x0000000B，即 11 个，接着开始的是字符串的内容，每个字符对应两个字节，因为是 11 个字符，因此一共有 22 个字节，值为 versionCode，在字符串末尾还有两个字节为字符串终止符，即 0x0000。根据此规则，27 个字符串的内容如表 1 所示，其中第 12 个字符串内容为空，在分析时需要注意。因为本文分析的 AndroidManifest 没有 Style 内容，所以 String Pool 结束后会进入 Resource ID 内容。

所谓的 Resource ID，即对应 Android 源码中/frameworks/base/core/res/res/values/public.xml 文件中每行的 id 值，如图 6 所示。图 7 为 ResourceId Chunk 的内容，开始处的 4 个字节内

表 1　27 个字符串内容

编号	字符串内容
0	versionCode
1	versionName
2	minSdkVersion
3	targetSdkVersion
4	name
5	allowBackup
6	icon
7	label
8	theme
9	android
10	http://schemas.android.com/apk/res/android
11	空字符串
12	package
13	manifest
14	com.android.test
15	1.0
16	uses-sdk
17	uses-permission
18	android.permission.INTERNET
19	application
20	activity
21	com.android.test.MainActivity
22	intent-filter
23	action
24	android.intent.action.MAIN
25	category
26	android.intent.category.LAUNCHER

```
28.  <public type="attr" name="theme" id="0x01010000" />
29.  <public type="attr" name="label" id="0x01010001" />
30.  <public type="attr" name="icon" id="0x01010002" />
31.  <public type="attr" name="name" id="0x01010003" />
32.  <public type="attr" name="manageSpaceActivity" id="0x01010004" />
33.  <public type="attr" name="allowClearUserData" id="0x01010005" />
34.  <public type="attr" name="permission" id="0x01010006" />
35.  <public type="attr" name="readPermission" id="0x01010007" />
36.  <public type="attr" name="writePermission" id="0x01010008" />
37.  <public type="attr" name="protectionLevel" id="0x01010009" />
38.  <public type="attr" name="permissionGroup" id="0x0101000a" />
39.  <public type="attr" name="sharedUserId" id="0x0101000b" />
40.  <public type="attr" name="hasCode" id="0x0101000c" />
41.  <public type="attr" name="persistent" id="0x0101000d" />
```

图 6 public.xml 文件部分内容

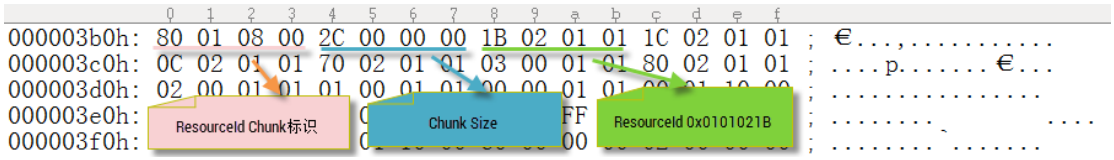

图 7 ResourceId Chunk

容为 0x00080180，即为 ResourcdId Chunk 的标识，接着的 4 个字节为本 Chunk 的大小，此处为 0x0000002C，即为 44 bytes，每个 ResouceId 值为 4 个字节，除去开头固定的 8 个字节，因此本 AndroidManifest 一共有 (44-4*2)/4 = 9 个 ResourceId。0x00003B08 开始的 4 个字节内容 0x0101021B 即为 ResourceId 值，根据 public.xml 文件内容，可知其对应值如下，但是在分析 AXMLPrinter2 的源码过程中发现 ResourceId 的值貌似没有使用到，ResourceId 对应的 name 在 String Pool 中有直接对应的值。

```
1.  <public type="attr" name="versionCode" id="0x0101021b" />
```

接下来即为 XML Content 的内容，此部分一共有 5 中类型的 Chunk，分别为 Start Namespace Chunk(0x00100100)，End Namespace Chunk(0x00100101)，Start Tag Chunk(0x00100102)，End Tag Chunk(0x00100103)以及 Text Chunk(0x00100104)。这 5 种 Chunk 的开头 4 个属性(16 bytes)的含义都是一样的，分别表示 Chunk 类型标识、Chunk Size，行号(对应编码前原文件内容的行号)、固定值 0xFFFFFFFF。

ResourceId Chunk 结束后，首先是 Start Namespace Chunk，类型标识为 0x00100100；大小为 0x00000018，即 24 bytes；行号为 0x00000002，对应编码前文件的第 2 行内容，即 <manifest xmlns:android=http://schemas.android.com/apk/res/android；Prefix 值为 0x00000009，其表示 String Pool 中的字符串索引值，即 android；Uri 值为 0x0000000A，其同样为 String Pool 中的字符串索引值，即 http://schemas.android.com/apk/res/android，Prefix 与 Uri 在此处是对应关系，后续的 Start Chunk Type 中的 Attribute 会根据 Uri 来获得 Prefix 值。

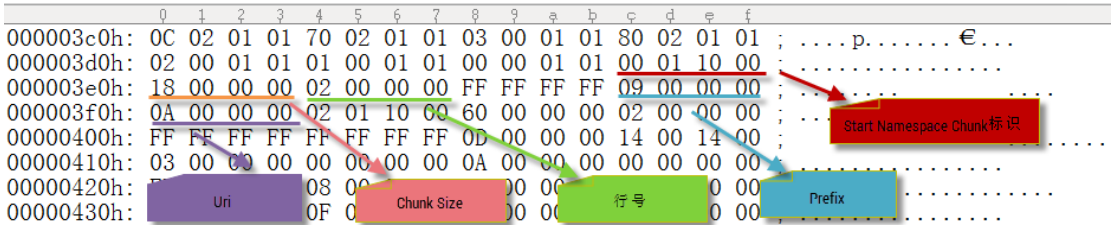

图 8 Start Namespace Chunk

Start Tag Chunk 的标识为 0x00100102；0x00000060 为 Chunk Size，即 96 bytes，因此其区间为 0x000003F4-0x00000453；0x00000404 处的 4 字节内容为 0xFFFFFFFF 为命名空间的 Uri，即为-1，根据 AXMLPrinter2 的源码当为-1 时，返回值为 null，命名空间为空；接下来的 4 个字节内容为该 Tag 的名称的字符串索引值 0x0000000D，因此值为 manifest；0x00000410 开始处的 4 个字节内容为 XML 标签下的属性个数，此处值为 0x00000003，即一共 3 个属性，

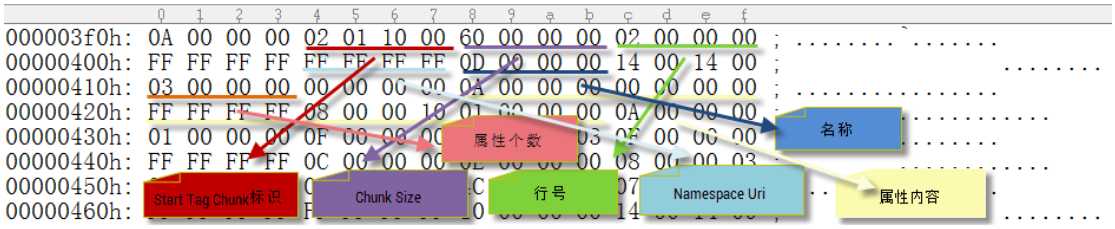

图 9 Start Tag Chunk

正好对应原文件内容中的 pacakge，android:versionCode，android:versionName 3 个属性。每个属性由 5*4 bytes 构成，实际上就是一个长度为 5 的数组，数组中每个值的含义为[Namespace Uri, Name, Value String, Type, Data]，继续以图 9 来分析。第一个属性的数组内容为[0x0000000A, 0x00000000, 0xFFFFFFFF, 0x10000008, 0x00000001]，而数组第 4 个值在实际处理过程中会首先右移 24 位(0x10000008>>>24)，因此数组值实际为[10, 0, -1, 16, 1]，第 1 个值为命名空间的 Uri，前面在 Start Namespace Chunk 中已经对应了 Prefix 和 Uri，因此 Uri 为 10 即对应到值为 9 的 Prefix，字符串内容为 android；第 2 个值为属性名称的字符串索引，因此值为 versionCode；第 3 个值在当 Type 为表示字符串时与 Data 值相同；第 4 个值表示数据的类型，包括 String、Attribute、Reference、Float、Int Hex、Int Boolean、Int 等(具体可以参考 AXMLPrinter2 的源码 test/AXMLPrinter 类中的 getAttributeValue 函数)，由于此处 Type 值为 16，表示是个整数值；第 5 个值即为属性的内容，由于 Type 指示为整数值，因此属性内容即为整数 1。综上，第一个属性解码后应为：android:VersionCode="1"。

其他几个 Chunk 比较简单，可以自行分析。