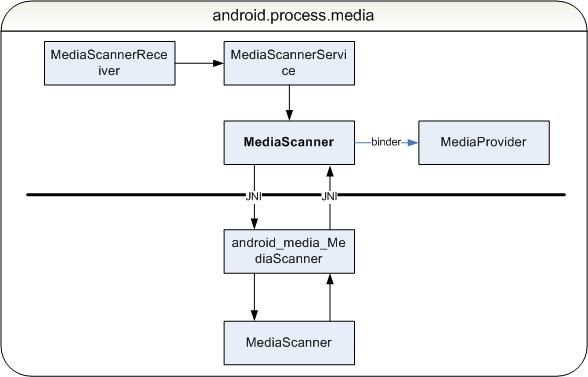
**Android 多媒体扫描过程（Android Media Scanner Process）**

下面是系统图   
        


MediaScannerReceiver会在任何的ACTION\_BOOT\_COMPLETED, ACTION\_MEDIA\_MOUNTED或 ACTION\_MEDIA\_SCANNER\_SCAN\_FILE 意图（intent）发出的时候启动。因为解析媒体文件的元数据或许会需要很长时间，所以MediaScannerReceiver会启动MediaScannerService。   
  
  
MediaScannerService调用一个公用类MediaScanner去处理真正的工作。MediaScannerReceiver维持两种扫描目录：一种是内部卷（internal volume）指向$(ANDROID\_ROOT)/media. 另一种是外部卷（external volume）指向$(EXTERNAL\_STORAGE).   
  
扫描和解析工作位于[Java](http://dev.10086.cn/cmdn/wiki/index.php?doc-view-1306)层和C++层。JAVA层是启动器。MediaScanner扫描所有目录，如下步骤：   
  
  
1.JAVA层初始化   
  
  
    在这一步骤中，它会根据目录是在内部卷还是外部卷打开不同的数据库。   
  
  
2.Java层预扫描   
  
  
    首先清除文件和播放列表的缓存条目。然后根据MediaProvider返回的请求结果生成新文件和播放列表缓存条目。   
  
  
3.C++层处理目录   
  
  
    列举出所有文件和特定的所有子目录（如果子目录包含一个.nomedia隐藏文件，则不会被列举出来。）。被列举的文件是根据文件扩展来判断文件是否被支持。如果支持这种文件扩展，C++层就会回调到JAVA层扫描文件。这种扩展就会被扫描到MediaFile.java中列出。下面是支持的文件扩展列表。   
  
  
/\* Audio \*/   
addFileType(&quot;MP3&quot;, FILE\_TYPE\_MP3, &quot;audio/mpeg&quot;);   
addFileType(&quot;M4A&quot;, FILE\_TYPE\_M4A, &quot;audio/mp4&quot;);   
addFileType(&quot;WAV&quot;, FILE\_TYPE\_WAV, &quot;audio/x-wav&quot;);   
addFileType(&quot;AMR&quot;, FILE\_TYPE\_AMR, &quot;audio/amr&quot;);   
addFileType(&quot;AWB&quot;, FILE\_TYPE\_AWB, &quot;audio/amr-wb&quot;);   
addFileType(&quot;WMA&quot;, FILE\_TYPE\_WMA, &quot;audio/x-ms-wma&quot;);   
addFileType(&quot;OGG&quot;, FILE\_TYPE\_OGG, &quot;application/ogg&quot;);   
addFileType(&quot;MID&quot;, FILE\_TYPE\_MID, &quot;audio/[MIDI](http://dev.10086.cn/cmdn/wiki/index.php?doc-view-1918)&quot;);   
addFileType(&quot;XMF&quot;, FILE\_TYPE\_MID, &quot;audio/midi&quot;);   
addFileType(&quot;RTTTL&quot;, FILE\_TYPE\_MID, &quot;audio/midi&quot;);   
addFileType(&quot;SMF&quot;, FILE\_TYPE\_SMF, &quot;audio/sp-midi&quot;);   
addFileType(&quot;IMY&quot;, FILE\_TYPE\_IMY, &quot;audio/imelody&quot;);   
  
/\* Video \*/   
addFileType(&quot;MP4&quot;, FILE\_TYPE\_MP4, &quot;video/mp4&quot;);   
addFileType(&quot;M4V&quot;, FILE\_TYPE\_M4V, &quot;video/mp4&quot;);   
addFileType(&quot;3GP&quot;, FILE\_TYPE\_3GPP, &quot;video/[3GPP](http://dev.10086.cn/cmdn/wiki/index.php?doc-view-1796)&quot;);   
addFileType(&quot;3GPP&quot;, FILE\_TYPE\_3GPP, &quot;video/3gpp&quot;);   
addFileType(&quot;3G2&quot;, FILE\_TYPE\_3GPP2, &quot;video/3gpp2&quot;);   
addFileType(&quot;3GPP2&quot;, FILE\_TYPE\_3GPP2, &quot;video/3gpp2&quot;);   
addFileType(&quot;WMV&quot;, FILE\_TYPE\_WMV, &quot;video/x-ms-wmv&quot;);   
  
/\* Image \*/   
addFileType(&quot;JPG&quot;, FILE\_TYPE\_JPEG, &quot;image/jpeg&quot;);   
addFileType(&quot;JPEG&quot;, FILE\_TYPE\_JPEG, &quot;image/jpeg&quot;);   
addFileType(&quot;GIF&quot;, FILE\_TYPE\_GIF, &quot;image/gif&quot;);   
addFileType(&quot;PNG&quot;, FILE\_TYPE\_PNG, &quot;image/png&quot;);   
addFileType(&quot;BMP&quot;, FILE\_TYPE\_BMP, &quot;image/x-ms-bmp&quot;);   
addFileType(&quot;WBMP&quot;, FILE\_TYPE\_WBMP, &quot;image/vnd.wap.wbmp&quot;);   
  
/\* Audio Play List \*/   
addFileType(&quot;M3U&quot;, FILE\_TYPE\_M3U, &quot;audio/x-mpegurl&quot;);   
addFileType(&quot;PLS&quot;, FILE\_TYPE\_PLS, &quot;audio/x-scpls&quot;);   
addFileType(&quot;WPL&quot;, FILE\_TYPE\_WPL, &quot;application/vnd.ms-wpl&quot;);   
  
  
4.Java层扫描文件   
  
  
  
    a）Java层开始文件   
  
  
  
首先它忽略一些MacOS 和 Windows Media Player特殊的文件。然后它会查看被扫描的文件是否已经存在于缓存条目中，如果存在，它会检查文件上次修改的时间是否改变。最后它返回该文件是否需要进一步处理的结果。如果不需要，接下来的两步不会执行。   
  
  
  
    b)C++层扫描文件   
  
  
  
不是所有的文件都需要交给C++层解析成元数据。只有下面的文件类型会被解析，注意，这里不处理image文件。

**Java代码**

1. **if** (mFileType == MediaFile.FILE\_TYPE\_MP3 ||
2. mFileType == MediaFile.FILE\_TYPE\_MP4 ||
3. mFileType == MediaFile.FILE\_TYPE\_M4A ||
4. mFileType == MediaFile.FILE\_TYPE\_3GPP ||
5. mFileType == MediaFile.FILE\_TYPE\_3GPP2 ||
6. mFileType == MediaFile.FILE\_TYPE\_OGG ||
7. mFileType == MediaFile.FILE\_TYPE\_MID ||
8. mFileType == MediaFile.FILE\_TYPE\_WMA) {

11. ……


15. ｝

复制代码   
  
对于被解析的元数据信息，C++层会回调到JAVA层的handleStringTag。Java层会记录它的name/value信息。   
  
  
    c)Java层结束文件   
  
  
   最后根据上一步解析出的值， Java层会更新相应的MeidaProvider产生的数据库表。   
  
  
5.Java层发送扫描   
  
  
    到目前为止，所有文件已经被扫描，它最后会检查文件和播放列表缓存条目，看是否所有项仍然存在于文件系统。如果有空条目，则会从数据库中删除。这样它能够保持数据库和文件系统的一致性。   
  
  
    其他的[应用程序](http://dev.10086.cn/cmdn/wiki/index.php?doc-view-1781)通过接收MediaScannerService发出的ACTION\_MEDIA\_SCANNER\_STARTED 和ACTION\_MEDIA\_SCANNER\_FINISHED意图能够知道什么时候扫描操作开始和结束。