广播现场直播智能化播出软件的实现

徐宇梁 季 强

摘 要： 在广播现场直播中，使用计算机对现场的实时声音信号进行短暂的缓存和简单的剪辑，避免重要的现场素材与台内广告发生档期冲突。软件设计时考虑了广播现场直播的特殊情况，使用了 java sound API，考虑了多平台的兼容并采用了多线程技术。

关键词： 广播现场直播；java 音频处理；java sound API；java 多线程

作者简介： 徐宇梁，男，工程师。（浙江广播电视集团 广播制播中心，浙江 杭州，310005）
季强，男，助工。（浙江广播电视集团 广播制播中心，浙江 杭州，310005）

中图分类号： TN939.1 **文献标识码：** **文章编号：** 1008-6552（2011）03-0098-03

一、引言

在目前广播现场直播中，普遍存在一个遗憾，现场的节目无法完整播出，经常会被广告打断。这个矛盾很难协调，主要有以下原因：广告的播出流程由电台直播室控制，直播室很难跟现场实时协调。某些广告已经签约为定点广告，无法改动播出的时间。直播现场必须照顾现场观众，不可能在广告时间暂停演出。

本项目使用了计算机将现场声音缓存、实时剪辑后播出，把现场最重要的内容呈现给听众，并能够删除一些不适合广播播出的内容，改善收听效果，提高节目质量。

二、系统结构

为了兼顾灵活与稳定，软件选择了适合多平台的 java 语言作为开发语言。目前可以在 Ubuntu Linux 10.10 和 windows xp 下运行。硬件选择了 Digigram UAX220 v2 声卡，这是一块便携式的 USB 声卡，可以方便地连接在笔记本计算机上。此声卡支持固定的 48KHz 采样率，能够在 windows linux 和 mac 三种平台下使用。

如图 1，计算机连接在播出平台的最后一级，直接连接传输设备（如 comrex）的输入端口，将计算机处理后的声音发送到台内，同时通过耳机监听 comrex 返送的台内信号。

图 1 硬件连接图

三、软件主要模块

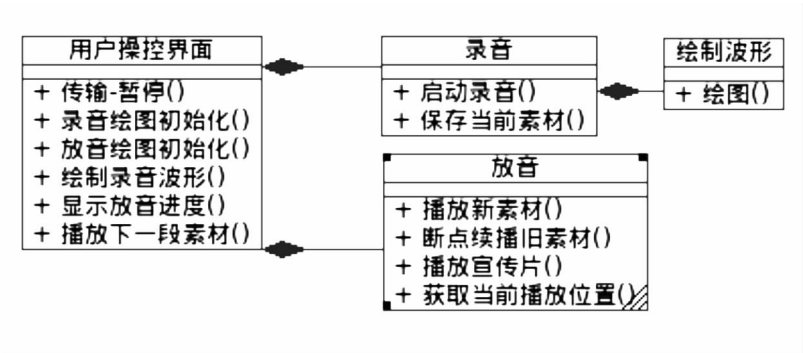

图 2　系统 UML 图

（一）主要结构

软件主要包括录音、放音、用户控制界面这三个模块，见图 2。三个模块都采用了多线程技术。因为在不间断录音的同时，我们还需要不间断地放音并操控素材。众所周知，协调多线程间的通讯比较复杂，将安全播出依赖于高超的编程技巧是不负责任的。本软件设计时坚持“KISS”原则，以保证程序可靠。在设计中，尽量简化线程间的通讯，录音线程只接受一个“存盘”指令。另外，录音这项工作客观上要求永不间断，所以按照规范先安全地退出线程，然后再存盘，或者接受一个“阻塞”指令，都是不允许的，因为这会打断录音。本软件简单设置了一个存盘函数，直接读取放置在内存中的声音数据，按照 java sound API 的要求转换成 wav 格式后直接存盘。

在放音模块中，则必须调度线程间的运作。因为 Java sound API 对于一段声音的播放设置了缓存，声音播放完毕，推出循环后，还需要调用 drain（）函数，才能把缓存中的声音放完。这造成一个后果，当主控线程以为声音已经播放完毕的时候，放音线程其实还在播放缓存中的声音。如果此时强行播放新的声音，就会造成漏播。另外，放音线程在理论上可以有多个，但是现实要求只能有一个，否则声音会叠加。结束当前线程，开始一个新线程的方法有很多，针对上面的情况，使用 thread. join（）方案是比较合适的，不会太着急打断前一个线程。

过多的线程协调会把程序逻辑搞成一团糟。一旦出现错误，多线程的排错调试又非常困难，所以本项目在用户控制界面这个模块中放弃使用多线程，转而使用计时器类（Timer），从另外两个线程中读取必要的数据显示给用户。计时器有多线程的效果，在逻辑上又非常清晰，因此是一个合适的选择。

（二）声音信号处理 Java Sound API

声音处理有很多种方法，大部分的多媒体程序都选择直接调用现成的控件。本项目放弃了这个方案，因为现成的控件有很多局限性。最大的局限在于，几乎所有的控件都不是线程安全的，另外也很难提供我们需要的图形绘制等功能。

java 声音处理有多个类，主要存放于 javax. sound. sampled 包中[1]。

简化后的录音代码：

```
AudioFormat. Encoding encoding = AudioFormat. Encoding. PCM_ SIGNED;
float rate = 48000f; //采样率
int sampleSize = 16; //采样深度
boolean bigEndian = true; //是否为 bigEndian
int channels = 2; //声道数
info [ ] mixerInfo = AudioSystem. getMixerInfo ( ); //混音器
```

```
line = (TargetDataLine) AudioSystem.getLine (info); //line 类用于读出声卡中的数据
line.open (format, line.getBufferSize ());
line.start ();
while (thread != null) {
if ( (numBytesRead = line.read (data, 0, bufferLengthInBytes)) == -1) {
break;
}
out.write (data, 0, numBytesRead); //把声卡上的数据写入一个输出流
}
```

声音的播放与此类似。

(三) 多平台下的问题

本项目采用的 Digigram UAX220 v2 声卡可以支持多平台，在 linux 下支持 alsa 驱动。实际使用的时候发现，由于 ubuntu linux 系统本身使用 pulse audio 管理声音，而不是直接使用 alsa，而 sun JDK 会强制使用 alsa 驱动，这导致系统无法方便地选择 Digigram 声卡为默认声卡，除非计算机里只有这一块声卡。笔记本一般都有集成声卡而且无法在 BIOS 中删除，这使得笔记本在安装了 ubuntu 和 sun JDK 后无法使用 Digigram 声卡。Open JDK 与 ubuntu linux 之间的兼容性比 Sun JDK 好很多，但是 Open JDK 与 Sun JDK 在 Sound API 实现上并不兼容，前者不支持 drain () 函数，只有调用 flush () 才能达到类似效果。Sun JDK 的声音播放能精确到微秒，而 Open JDK 只能精确到毫秒。

(四) java 的内存溢出

作为一个多媒体软件，对内存的占用相当大，必须考虑内存是否有溢出。内存泄露对于播出系统而言是致命的。通常，在 java 语言中，new 一个对象后，并不需要显式地 delete 它，垃圾处理器会自动择机释放它。但是实际使用中发现，不能过于信赖 java 的自动垃圾处理机制。本项目中，录音模块是一个很适合采用“单例模式”的模块，java 的垃圾处理机制一旦碰到“单例模式”，它会默认所有的数据都需要一直使用下去，不会主动释放内存。录音时暂时存放在内存中的声音数据会一直积累下去，直到崩溃。

在编写本软件过程中，还发现了其他几种内存泄露现象。从原理上讲，java 的垃圾收集器平时会监控每个对象的申请、引用、赋值..... 一旦发现某对象不再被引用，就会择机释放它的内存。垃圾收集器采用的是有向图，即便某几个对象互相引用，只要它们与根进程处于不可达的状态，它们也能被释放。因此，当声音数据被放入一个容器类的时候，如果不在容器中删除它，仅仅把它自身删除，也是没有用的，因为容器还在引用它。不正确的使用一些图形界面控件也会造成这个问题。比如 javax.swing.JSlider 控件就会引用一个 SmartHashtable。

四、小　　结

开发广播现场直播智能播出软件，是为了“美化”现场直播节目，提高实际播出节目现场部分的完整性和可听性。本系统采用计算机做全程录音，短暂延时后播放，延时会影响“直播”的基本特质，即现场扩声和广播播出的内容不同步，同时也要求电台转播必须比现场稍微晚一点点，这都需要对目前的直播习惯作一些修改。但是这个短暂的延时也带来了前所未有的好处，不但能缓解现场信号与台内广告的冲突，还给现场操作者提供了缓冲余地，方便地“掩饰”现场突发的噪声、静音或者短暂的线路故障。

参考文献：

[1] Oracle. Java? Platform, Standard Edition 6 API Specification javax.sound.sampled 部分 http://download.oracle.com/javase/6/docs/api/, 2010.