doi: 10.3969/j.issn.1006-1576.2010.03.032

Linux 下语音实时通信的一种实现方法

程虹霞[1]，朱珠[2]，吴小强[3]，龙建忠[3]
（1. 中国兵器工业第五八研究所 军品部，四川 绵阳 621000；
2. 中国兵器工业第五八研究所 国防科技工业弹药自动装药技术研究应用中心管理办公室，四川 绵阳 621000；
3. 四川大学 电子信息学院，四川 成都 610064）

摘要：在设计嵌入式语音通信系统的过程中，提出了一种 Linux 下语音实时通信方法。利用 Linux 下 OSS 内核驱动实现了语音的采集和回放，基于 LIVE 库实现了 RTP/RTCP 的音频流的实时传输，并运用 Linux 下多线程技术，实现了语音的全双工数字通信。调试结果表明，该方法可以快速方便的实现语音的实时传输，对从事相关产品的开发具有一定的参考价值。

关键词：Linux；OSS；RTP/RTCP；LIVE；多线程技术
中图分类号：TP311.52 **文献标识码**：A

An Approach of Real-Time Speech Communication Based on Linux

CHENG Hong-xia[1], ZHU Zhu[2], WU Xiao-qiang[3], LONG Jian-zhong[3]
(1. Dept. of Armament Products, No. 58 Research Institute of China Ordnance Industries, Mianyang 621000, China;
2. Management Office, Research & Application Center for Ammunition Automatic Charging & Assembly of National Defense Science & Technology Industry, No. 58 Research institute of China Ordnance Industries, Mianyang 621000, China;
3. College of Electric Information, Sichuan University, Chengdu 610064, China)

Abstract: An approach was proposed that real-time speech communication based on Linux system during the period of embedded speech communication system designing. It has adopted the technology of OSS to complete the voice collection and replay, and realize the real-time transmission based on RTP/RTCP by the medium of LIVE storeroom, at the same time, it has made use of multithread technology to accomplish the speech communication on the full duplex. The experiments verify that the approach proposed above is really feasible and efficient. This approach will serve as references to the designs of the correlation products.

Keywords: Linux; OSS; RTP/RTCP; LIVE; Multi- thread technology

0 引言

Linux 是目前嵌入式领域中应用广泛的操作系统，其产品线很长，可根据硬件设计、应用领域、计算环境甚至用户的兴趣爱好，提供特定的产品和解决方案；另外，Linux 源码开放，完全免费，具有高度可靠的先进性、安全性和稳定性，相对于价格不菲且源码封装的其他商业操作系统来说，在 Linux 下开发嵌入式相关产品，对用户来说是一种更为实际的选择。故在设计嵌入式语音通信系统的过程中，提出一种 Linux 下语音实时通信的方法。

1 Linux 下 OSS 音频编程方法

开放式语音系统（Open Sound System，OSS）以内核驱动程序的形式在 Linux 内核空间运行，出于对安全性方面的考虑，Linux 下的应用程序无法直接对声卡这类硬件设备进行操作，必须通过内核提供的驱动程序才能完成。只要音频处理应用程序按照 OSS 的 API 来编写，在移植到另外一个平台时，只需要重新编译即可。应用程序、内核（声卡）驱动程序和硬件资源（声卡）的关系[1]如图 1。

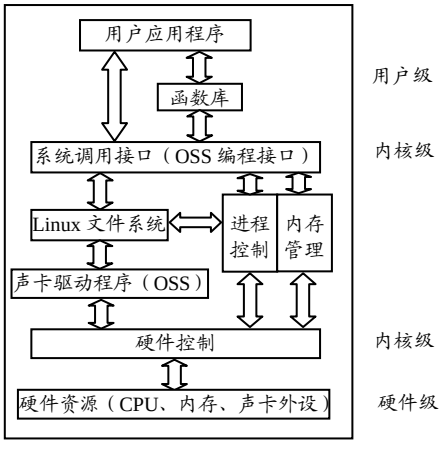

图 1 应用程序、内核驱动程序和硬件资源的关系

OSS 为音频编程提供 3 种设备[2]，分别是/dev/dsp，/dev/dspW 和/dev/audio，读写这个设

收稿日期：2009-10-21；修回日期：2009-11-25
基金项目：企事业委托项目《基于 TCP/IP 网络的小区智能终端研制》（04H700）
作者简介：程虹霞（1975-），女，安徽人，工学硕士，毕业于四川大学电子信息学院，从事武器装备总体论证研究。

备相当于录音和放音。/dev/dsp 与/dev/audio 之间的区别在于采样的编码不同，/dev/audio 使用 μ 律编码，/dev/dsp 使用 8-bit（无符号）线性编码，/dev/dspW 使用 16-bit（有符号）线形编码，/dev/audio 主要是为了与 SunOS 兼容。通常对/dev/dsp 编程实现音频设备的录放音功能。

1) 打开音频设备

定义头文件后，用 open 系统调用建立起与硬件间的联系：

```
int open(const char */dev/dsp，int O_RDWR);
```

此时返回的文件描述符将作为随后操作的标识。open_mode 有：O_RDONLY、O_WRONLY 和 O_RDWR 选择，分别表示只读、只写和读写，全双工的情况下（即录音和放音同时）使用读写模式。

2) 录音/放音

使用 read/write 系统调用从设备接收数据或向设备写入数据：

```
int read(int fd，char *buf，size_t count);
size_t write(int fd，  const char *buf，  size_t count);
```

其它不符合读/写基本模式的操作由 ioctl 系统调用完成。录音/放音时必须读/写一个完整的采样。如一个 16-bit 的立体声模式下，每个采样有 4 个字节，故应用程序每次必须读/写 4 的倍数个字节。另外，由于 OSS 是一个跨平台的音频接口，编程时要考虑到可移植性的问题，注意读/写时的字节顺序。

3) 设置参数

```
//设置采样格式
int format;
format = AFMT_S16_LE;
if (ioctl(audio_fd, SNDCTL_DSP_SETFMT, &format) == -1) {
perror("SNDCTL_DSP_SETFMT");
exit(1);
}
//设置通道数目
int channels = 2; /* 1=mono, 2=stereo */
if(ioctl(audio_fd, SNDCTL_DSP_CHANNELS, &channels) == -1)
{
perror("SNDCTL_DSP_CHANNELS");
exit(1);
}
//设置采样速率
int speed = 8000;
if (ioctl(audio_fd, SNDCTL_DSP_SPEED, &speed)==-1) {
perror("SNDCTL_DSP_SPEED");
exit(Error code);
}
```

2　基于 LIVE 库的 RTP/RTCP 实时音频流传输设计

实时传输协议[4-5]（Real-time Transport Protocol, RTP）是用于 Internet 上针对多媒体数据流的一种传输协议。RTP 被定义为在一对一或一对多的传输下工作，其目的是提供时间信息和实现流同步。RTP 通常使用 UDP 来传送数据，也可在 TCP 或 ATM 等其他协议之上工作。当应用程序开始一个 RTP 会话时将使用 2 个端口：一个给 RTP，一个给 RTCP。RTP 并不能为按顺序传送数据包提供可靠的传送机制，也不提供流量控制或拥塞控制，它依靠 RTCP 提供这些服务。通常 RTP 算法并不作为一个独立的网络层来实现，而是作为应用程序代码的一部分。

实时传输控制协议 RTCP[4-5]（Real-time Transport Control Protocol）和 RTP 一起提供流量控制和拥塞控制服务。在 RTP 会话期间，各参与者周期性地传送 RTCP 包。RTCP 包中含有已发送的数据包的数量、丢失的数据包的数量等统计资料。服务器可利用这些信息动态地改变传输速率，甚至改变有效载荷类型。RTP 和 RTCP 配合使用，它们能以有效的反馈和最小的开销使传输效率最佳化，因而特别适合传送网上的实时数据。

在 Linux 平台上进行实时流媒体编程，通常考虑使用一些开放源代码的 RTP 库，如 LIBRTP、JRTPLIB、LIVE 等。LIVE 库是使用开放的标准协议（RTP/RTCP，RTSP，SIP）用于流媒体传输的 C++库，可用以构建基本的 RTSP 或 SIP 客户端和服务器端。LIVE 库让用户使用 RTP 协议发送和接受数据变为可能，而不用考虑 SSRC 冲突，调度和传输 RTCP 数据等，RTPsession 类提供了发送接受 RTP 包所必须的函数，通过创建一个 RTP 会话应用实例对 RTCP 包进行自动处理。

会话时，由主程序 talk.cpp 首先创建一个 RTPSession 类的一个实例来表示此次 RTP 会话，然后调用该类的 Create()方法来对其进行初始化操作。RTPSession 类的 Create()方法的个参数，用来指明此次 RTP 会话所采用的端口号。

```
RTPSession talk;   talk.Create(5000);
```

初始化时，RTPSession 类的 SetDefaultPayloadType()、SetDefaultMark()和 SetDefaultTimeStampIncrement ()方法来设置负载类型、标识和时戳增量。完成会话参数设置后，RTPUDPv4TransmissionParams 类的 SetPortbase ()方法来设置端口等 RTP 传输参数。

当 RTP 会话成功建立起来之后，开始进行音频流数据的实时传输。首先需要设置好数据发送的目

标地址，RTP 协议允许同一会话存在多个目标地址，可通过调用 RTPSession 类的 AddDestination()、DeleteDestination()和 ClearDestinations()来完成。

AddDestination()方法来指定音频流的目标地址；同时需申请动态的音频输入缓冲区 buf []，函数 int read(int fd，char*buf，size_t count)从声卡中读取数据，保存在 buf 字符指针所指向的缓冲区内。

目标地址全部指定之后，再调用 RTPSession 类的 SendPacket()方法，向所有的目标地址发送流媒体数据，SendPacket()是 RTPSession 类提供的一个重载函数。对于同一个 RTP 会话来讲，负载类型、标识和时戳增量通常来讲都是相同的，LIVE 允许将它们设置为会话的默认参数，该设置通过调用 RTPSession 类的 SetDefaultPayloadType()、SetDefaultMark()和 SetDefaultTimeStampIncrement()方法来完成。为 RTP 会话设置这些默认参数的好处是可以简化数据的发送，最终通过 SendPacket()方法向所有的目标地址发送流媒体数据。

3 基于多线程的语音全双工设计

语音实时通信方案采用 C/S 模式开发，分别设计基于 TCP/IP 的以太网相连的服务器端程序和客户端程序。服务器端用于发送音频流，客户端接收音频流，通过运用 Linux 下多线程技术[5-6]来实现语音的全双工数字通信。

服务器端程序中定义函数 AudioStreamer()和结构体 sessionState_t，AudioStreamer()利用 play()函数负责整个程序 RTP 和 RTCP 会话的创建和实现，服务器端程序在整个系统中作为主程序 talk.cpp 中的一个线程来实现。客户端的设计和服务器端类似，除了必要的接收对话的设置和 RTP/RTCP 设置外，需要在本地开辟缓冲区，进行语音的编解码和 μ 律到线性 PCM 的转换，使处理好后的语音供声卡读取。缓冲区的设置使用循环队列的方式，以避免普通堆栈“假上溢”的缺点。

主程序用 pthread_create()函数创建专用语音线程 int pthread_create(pthread_t *tid, const pthread_attr_t *attr, void *(* func) (void *), void *arg);，利用线程调用，分别启动客户端和服务器端程序；程序运行过程中，需要等待一个给定线程终止 int pthread_join(pthread_t tid, void **status);同时，得到自身的 pid，pthread_t pthread_self(void)。

pthread_detach 函数把指定的线程转变为脱离状态，一个线程或是可汇合的（joinable，缺省值），或是脱离的（detached），当一个可汇合的线程终止时，线程 ID 和退出状态将留到另个线程对它调用 pthread_join。脱离线程终止时相关资源都被释放。

使用互斥锁来保证一段时间内只有一个线程在执行一段代码，以使各个线程不同时向同一个文件顺序写入数据。使用 int pthread_mutex_lock 锁住互斥锁，使用 int pthread_mutex_unlock 解锁。如果向为一个已被其他线程锁住的互斥锁加锁，程序便会阻塞直到该互斥对象解锁。如果在共享内存中分配一个互斥锁，需要运行时调用 ptgread_mutex_init 函数初始化，pthread_cond_broadcast 用于设置条件变量，即使事件发生，并使得所有等待该事件的线程不再阻塞。同时用 pthread_cond_signal 来解除某一个等待线程的阻塞状态。

用 pthread_attr_setscope()函数设置线程绑定状态。它有 2 个参数，第 1 个是指向属性结构的指针，第 2 个是绑定类型，它有 2 个取值：PTHREAD_SCOPE_SYSTEM（绑定的）和 PTHREAD_SCOPE_PROCESS（非绑定的）。

```
pthread_attr_setscope(&attr,PTHREAD_SCOPE_SYSTEM);
pthread_create(&tid, &attr, (void *) talk, NULL);
```

线程完毕，用 pthread_cond_destroy 用来释放一个条件变量的资源。

4 结论

该设计可在局域网的点对点传输中进行语音通信，且语音连续性好、时延小、回声小。

参考文献：

[1] 晨风. 嵌入式实时多任务软件开发基础[M]. 北京：清华大学出版社, 2004.

[2] Dapeng Wu, Yiwei Thoms Hou, transporting Real-Time Video over the Internet[J]. Challenges and Approaches, Proceeding of the IEEE. 2000, 88(12): 1855-1874.

[3] Douglas E.Comer, David L.Stevens InternetWorking with TCP/IP Volume 3 Client-Server Programming and Application Linux/POSIX SOCKET VERSION[M]. 北京：人民邮电出版社, 2004.

[4] H.Schulzrinne, S.Casner, R.Frederick, and. Jacobson, “RTP: A transport protocol for real-time applicaitions”, IRTF Audio/Video Transport Working Group, January 1996.RFC1889

[5] Douglas E. Comer David L. Stevens. TCP/IP 网络互联技术（卷 3）—客户-服务器编程与应用（Windows 套接字版）[M]. 张卫，王能，译. 北京：清华大学出版社, 2004.

[6] 张斌. Linux2.6.16 在嵌入式模块 X-Board（GP8）上的移植[J]. 兵工自动化, 2009, 28(10): 94-96.