用 DirectSound 实现声音的实时采集、处理与播放

汤春林，左 勇，张传华
（中国空气动力研究与发展中心通信修理所，四川 绵阳 621000）

摘 要：DirectSound 应用编程接口是 DirectX 应用编程接口的一个组件。文章介绍了 DirectSound 的基本原理，并就使用 DirectSound 在计算机网络中实时捕获、处理和播放声音进行了介绍和探讨。

关键词：DirectSound；声音捕获；实时传输；回放
中图分类号：TN912 **文献标识码**：A **文章编号**：1005－3751(2003)0112－03

Using DirectSound to Capture, Process and Play Sounds in Real－time System

TANG Chun-lin, ZUO Yong, ZHANG Chuan-hua
(Communication Equipment Repair Station, CDRC, Mianyang 621000, China)

Abstract: The DirectSound API is the wave－audio component of the DirectX API. Here introduces the basic principle of DirectSound and the method of using it to capture, process and playback sounds in real－time computer networks.

Key words: DirectSound; capture sounds; real－time transmit; playback

0 引 言

声音是一种最直接、最方便进行信息交流的手段，在我们研制风洞试验指挥通信系统时，如何利用声卡对语音进行实时捕获、回放、混音，以及如何在网络中实时传递语音成为首先要解决的问题。在这方面我们采用微软的 DirectX SDk(软件开发包)。DirectX 是一个用于多媒体应用程序和硬件增强的编程环境，每个 DirectX 组件都是用户可调用的 API(application programming interface)的总和，应用程序使用它可以直接访问计算机的硬件。DirectSound API 是 DirectX API 的音频组件，它提供低层的混音、硬件加速和音频设备的直接访问。利用 DirectSound API 提供的函数，可以方便地进行声音的实时采集、处理、传输和回放。

1 DirectSound 基本原理

DirectSound 以 COM(Component Object Model，组件对象模型)形式存在，C++可以通过接口指针调用接口提供的函数来实现声音数据的捕获、处理和回放。

应用 DirectSound 主要是应用 DirectSound 回放和 DirectSound 捕获。DirectSound 回放基于 IDirectSound COM 接口和处理声音缓存和 3－D 效果的 COM 接口，这些接口是 IDirectSoundBuffer，IDirectSound3Dbuffer 和 IDirectSound3Dlistener。DirectSound 捕获基于 IDirectSoundCapture 和 IDirectSoundCaptureBuffer COM 接口。为使 DirectSound 回放和 DirectSound 捕获的缓存指针到达一定点后有事件发出，二者都有 IDirectSoundNotify 接口来实现此功能。

1.1 DirectSound 数据块结构

IDirectSound 和 IDirectSoundCapture 以波形声音数据方式工作，这些数据是固定频率下的采样。数据块结构由 WAVEFORMATEX 描述，其结构声明如下：

```
typedef struct {
    WORD wFormatTag;
    WORD nChannels;
    DWORD nSamplesPerSec;
    DWORD nAvgBytesPerSec;
    WORD nBlockAlign;
    WORD wBitsPerSample;
    WORD cbSize;
} WAVEFORMATEX;
```

wFormatTag 为格式标识，对于 DirectSound 只能是 WAVE_FORMAT_PCM，表示播放(捕获)是 PCM(Pulse Code Modulation)格式；nChannels 表示通道数，单声道为 1，立体声为 2；nSamplesPerSec 表示采样频率，一般为8000 Hz、11 025 Hz、22 050 Hz 和 44 100 Hz；wBitsPerSample 表示采样量化的比特数，通常是 8 或 16；nBlockAlign 表示采样块的字节数，对于 PCM 是 wBitsPerSample * nChannels / 8；nAvgBytesPerSec 表示采样率，以字节为单位，由

收稿日期：2003－04－18
作者简介：汤春林(1971—)，男，江苏南京人，硕士，工程师，研究方向为多媒体通信技术。

nBlockAlign 和 nSamplesPerSec 决定;cbSize 表示描述特定波形格式所需的额外尺寸,对于 PCM 格式此参数为 0。

1.2 DirectSound 回放

DirectSound 回放缓存用于控制回放声音的开始、停止、暂停,回放声音时声音数据存放在 DirectSound 缓存中。DirectSound 回放缓存分为主缓存(primary sound buffer)和二级缓存(secondary sound buffer)。主缓存只有一个,直接与硬件相关,用于存放待播放的数据。二级缓存可以有多个,一个二级缓存存放一路播放数据,二级缓存的数据只有在主缓存上才能播放。各二级缓存的播放数据在主缓存混合,由主缓存送到音频输出设备,这就可以方便地实现混音。在创建 DirectSound 时即创建了主缓存,应用程序需设置其数据块格式;二级缓存由应用程序自己创建,应用程序只需把数据放入二级缓存就可以播放了。

通过 DirectSound 的 IDirectSoundNotify 组件接口,DirectSound 可以支持流方式播放,应用程序可以通过 IDirectSoundNotify 在二级缓存中设置若干通知点,当播放指针到达通知点时就发出到达事件,应用程序就可以填充一段新的数据到二级缓存。

1.3 DirectSound 捕获

DirectSound 捕获应用 DirectSoundCapture 对象。捕获缓存用于存放从音频输入设备捕获来的音频数据,在创建 DirectSoundCapture 时即创建捕获缓存。捕获缓存是循环缓存,当捕获指针到达缓存末尾时,指针又从缓存的头部开始,因此捕获的数据要及时取出。通过 IDirectSoundNotify 接口,可以设置通知点,在捕获缓存指针到达通知点时即发出一个事件,应用程序就可以取出捕获的声音数据。

2 用 DirectSound 实现声音实时捕获、处理与回放

在 IP 网络中捕获、处理和回放声音与电路交换的通信网络相比最大的不同是 IP 网络是虚电路,数据以 IP 包方式传递,具有时延不确定性,而且作为实时系统,在传输发生错误时不能使用 ARQ 方式重发,因此捕获的语音数据包不能太大,而且处理必须及时。下面分别就捕获、处理和回放进行论述。

2.1 初始化 DirectSound

在指挥通信系统中声音捕获和回放需要同时进行,所以 DirectSound 对象和 DirectSoundCapture 对象均需创建,在创建顺序上 DirectSound 规定必须先创建 DirectSound 对象。设置回放主缓存的数据格式也在初始化 DirectSound 中处理。要初始化成功,声卡必须支持双 DMA 通道。初始化的主要代码如下。

```
CoInitialize(NULL);// 初始化 COM
DirectSoundCreate( NULL, &g.pDS, NULL ); // 用主音频设备创建 IDirectSound
DirectSoundCaptureCreate( NULL, &g.pDSCapture, NULL );// 创建 IDirectSoundCapture
g.pDS→CreateSoundBuffer( &dsbd, &pDS BPrimary, NULL );// 创建回放的主缓存
// 设置主缓存数据块格式为 22kHz、16 - bit
WAVEFORMATEX wfx;
ZeroMemory( &wfx, sizeof(WAVEFORMATEX) );
wfx. wFormatTag = WAVE_FORMAT_PCM;
wfx. nChannels = 2;
wfx. nSamplesPerSec = 22050;
wfx. wBitsPerSample = 16;
wfx. nBlockAlign = wfx. wBitsPerSample / 8 * wfx. nChannels;
wfx. nAvgBytesPerSec = wfx. nSamplesPerSec * wfx. nBlockAlign;
pDSBPrimary→SetFormat(&wfx);
```

在上述代码中 g-pDS 为 LPDIRECTSOUND 指针,g-pDSCapture 为 LPDIRECTSOUNDCAPTURE 指针,pDSBPrimary 为 LPDIRECTSOUNDBUFFER 指针,dsbd 为 DSBUFFERDESC 结构。

2.2 创建捕获缓存

捕获缓存是循环缓存,在指挥通信系统中捕获缓存设 16 个通知点,每通知点 512 字节,这是兼顾实时性和音质而采用的参数,在不同的应用中也可以用不同的参数。主要代码如下:

```
g.dwNotifySizeCapture = 512; // 设置通知点大小
g.dwCaptureBufferSize = g.dwNotifySizeCapture * 16; // 设置捕获缓存为 512 * 16
WAVEFORMATEX g.wfxInput; // 设置捕获缓存数据块格式为 8kHz、8 - bit,具体略
// 创建捕获缓存
DSCBUFFERDESC dscbd;
ZeroMemory( &dscbd, sizeof(dscbd) );
dscbd. dwSize = sizeof(dscbd);
dscbd. dwBufferBytes = g.dwCaptureBufferSize;
dscbd. lpwfxFormat = pwfxInput; // Set the format during creatation
g.pDSCapture → CreateCaptureBuffer ( &dscbd, &g.pDSBCapture, NULL );
// 用 IdirectSoundNotify 接口设置捕获缓存的通知点
g.pDSBCapture → QueryInterface ( IID_IDirectSoundNotify, (VOID * * )&g.pDSNotifyCapture );
// 设置通知点位置
for( int i = 0; i < 16; i+ + ) {
g.aPosNotifyCapture[i]. dwOffset = (g-dwNotifySizeCapture * i) + g.dwNotifySizeCapture - 1;
```

```
g.aPosNotifyCapture [ i ]. hEventNotify = g.hNotifi-
cationEventsCapture[0];
}
g.aPosNotifyCapture[i]. dwOffset = DSBPN.OFFSET-
STOP;
g.aPosNotifyCapture [ i ]. hEventNotify = g.hNotifi-
cationEventsCapture[1];
g.pDSNotifyCapture→SetNotificationPositions(17, g.a-
PosNotifyCapture );
```

2.3 创建回放二级缓存

通信一旦开始,来自网络IP包的语音数据就源源不断送到回放端回放,因此回放二级缓存必须为循环缓存,即二级缓存的播放方式设为DSBPLAY.LOOPING(循环)方式。由于是双工实时声音捕获、处理和回放,声音捕获/发送和声音接收/播放是同时进行而且速率相同,因此在捕获端设有通知点,回放端不设通知点,新数据到来后即填入回放二级缓存。主要代码如下。

```
//设置回放二级缓存数据块格式与捕获缓存数据块
格式相同,即8kHz、8-bit,具体略
AVEFORMATEX m.pwfx;
g.dwBufferSize = 3072;//设置二级缓存为3 072字
节
//设置回放二级缓存
DSBUFFERDESC dsbd;
ZeroMemory( &dsbd, sizeof(DSBUFFERDESC) );
dsbd. dwSize = sizeof(DSBUFFERDESC);
dsbd. dwFlags=DSBCAPS.CTRLPOSITIONNOTIFY|
DSBCAPS.GETCURRENTPOSITION2;
dsbd. dwBufferBytes = g.dwBufferSize;
dsbd. lpwfxFormat = &m.pwfx;
CreateSoundBuffer( &dsbd, &g.pDSBuffer, NULL );
```

代码中g.pDSBuffer为二级缓存的LPDIRECTSOUNDBUFFER指针,多路混音时需创建多个二级缓存指针,各指针分别操作。

2.4 取捕获的声音数据

在捕获缓存中,捕获到的声音数据到达一个通知点即发出一个事件通知数据打包与传输线程取数,数据打包与传输线程通过调用捕获缓存接口的Lock()函数获得待取声音数据的指针和数据长度。利用Lock()函数输出的这两个参数就可以从捕获缓存中取出数据,然后对数据分包传输。需要指出的是虽然捕获缓存的通知点是512字节,但由于是多线程工作,声卡捕获到的数据在捕获缓存中过一个通知点时而捕获线程未获得CPU时间片,则不能发出通知事件,所以Lock()函数输出的数据长度有时不是512字节,而是512的倍数,所以数据打包与传输线程要根据实际的数据长度处理。取捕获缓存中声音数据的主要代码如下。

```
g.pDSBCapture→GetCurrentPosition( &dw Capture-
Pos, &dwReadPos );//获取当前的读偏移量
lLockSize = dwReadPos - g.dwNextCaptureOffset;//
获取已捕获而未读出的数据长度量
if(lLockSize<0)lLockSize+ =g.dwCaptureBufferSize;
lLockSize - = (lLockSize % g.dwNotifySizeCapture);
//锁定捕获缓存lLockSize长度,输出可以取出数据
的指针和实际可以取出数据的长度
g.pDSBCapture → Lock ( g.dwNextCaptureOffset,
lLockSize, &pbCaptureData,
&dwCaptureLength, NULL, NULL, 0L );
* dwCaptureDataLength = dwCaptureLength;//输出
捕获的数据长度
memcpy(pTempCaptureData,(BYTE * )pbCaptureDa-
ta,dwCaptureLength);//输出捕获的声音数据
g.pDSBCapture→ Unlock (pbCaptureData, dwCapture-
Length,NULL,0 );//捕获缓存解锁
g.dwNextCaptureOffset + = dwCaptureLength; // 改
变捕获缓存的取数据指针
g.dwNextCaptureOffset% = g.dwCaptureBufferSize; //
因是循环缓存
```

2.5 向播放缓存填充声音数据

数据接收与解包线程接收到声音数据后,根据上次填充数据获得的回放二级缓存的写数偏移量调用回放二级缓存接口的Lock()函数,获得可写数据的起始位置和实际可写的长度后即可以向回放二级缓存中填充数据。填入回放二级缓存中的数据由DirectSound自行控制送入回放主缓存中播放。主要代码如下。

```
//锁定回放二级缓存dwWantWriteSize长度,输出可
以可写数据的起始位置和实际可写的长度
g.pDSBuffer → Lock ( dwNextWriteOffsetIn,
dwWantWriteSize,&pbBuffer,
&dwBufferLength,NULL,NULL,0L );
memcpy((BYTE * )pbBuffer,pJxwPlaySoundData,dw-
BufferLength);//拷贝声音数据到指定位置
g.pDSBuffer → Unlock ( pbBuffer, dwBufferLength,
NULL,0); // 回放二级缓存解锁
dwNextWriteOffsetIn + = dwBufferLength; // 改变
回放二级缓存的写数据指针
dwNextWriteOffsetIn % = g.dwBufferSize; //因是循
环缓存.
* dwNextWriteOffsetOut = dwNextWriteOffsetIn;//
输出回放二级缓存的写数据指针
```

2.6 需要说明的问题

DirectSound与硬件有关,在使用DirectSound时创建

(下转第118页)

本文除文件系统外，不讨论其它的单一系统映像层，而研究其中的三个核心问题：控制器与存储设备、控制器与控制器以及系统与用户间的接口与通讯、管理存储设备的软件 RAID 技术和共享存储设备所带来的共享冲突问题的解决方法。

3 结束语

网络存储技术是近年来国际学术界和产业界的热点，其中所取得的任何成果对于存储系统性能和可用性的提升、对于高速网络服务器整体性能的提升都具有重要的意义。文中综合研究了网络存储中附网存储、存储区域网络和 ISCSI 三种技术的基本原理、结构及其问题，并基于光纤通道、RAID 和集群技术，提出一种高可用数据中心的体系结构。我们对其中的设备接口、容灾技术、共享冲突的解决方法等核心问题进行深入探讨。本文进行了如下主要工作并所获得的重要结论为：

(1) 网络存储技术通过网络的开放性和可扩展性，能够较好地满足存储系统的容量与性能日益增长的需要，可用性则可通过集群技术得到彻底解决。

(2) 深入研究目前比较流行的几种网络存储的体系结构及其接口技术，并详细分析了附网存储、存储局域网和 ISCSI 这三种存储技术的结构以及它们各自的优缺点。

(3) 研究了传统的容灾系统中网络数据访问的方法及其缺陷。

(4) 提出了一种构造高可用性数据中心的详细的设计方案，并对存储数据的 I/O 操作提出了相应的优化方法，以及给出了提高系统可用性的具体方法。

参考文献：

[1] Anderson D. Network Attached storage is inevitable[A]. Proc. of the 30th Hawaii International Conf. On System Sciences [C]. Hawaii:[s. n.],1997. 79 - 84.

[2] Clark T. Designing Storage Area Network[M][s. l.]: Addison - Wesley LongMan Inc,1999. 47 - 86.

[3] IETF, ISCSI standard [EB/OL]. http://www. ietf. org/html. charters/ips - charter. html,2001 - 10.

[4] Lee E K. Highly - Available, Scalable Network Storage[A]. 1995 Spring COMPCON[C]. [s. l.]:[s. n.],1995. 41 - 48.

[5] Artecon Corp. Technical Brief, SAN, NAS, and Direct - Attached Storage: What's Right for Your Network? [EB/OL]. http://www. artecon. com, 2002 - 01.

[6] Jurgens C. Fibre Channel: A connection to the future[J]. IEEE Computer, 1995,(8): 82 - 90.

[7] Farley M. SAN 存储区域网络[M]. 孙功星等译. 北京：机械工业出版社，2002. 205 - 405.

[8] 李丽娜. 网络存储——IT 的第 3 次浪潮[N]. 北京日报，2001 - 03 - 01.

[9] 张旭萍. 信息存储技术[M]. 北京：电子工业出版社，2001.

(上接第 111 页)

一起使用。

另外还须编制文件 NET. CFG，内容如下：

```
link support
    Max Stacks 8
Link Driver NE2000
    Int 3
    Port 300
    Frame Ethernet-802.3
    Frame Ethernet-II
    Frame Ethernet-SNAP
    Protocol IPX 0 Ethernet-802.3
```

其中 Int 和 Port 分别代表网卡的中断号和网卡地址。

6 结束语

用以太网代替串行口进行通讯是必然的趋势。设备的改造有多种方式，利用 DOS 本身提供的 Novell NetWare 低层 API 功能来实现 IPX 通讯是一个简便易行的办法，依托于 DOS 这个实时操作系统，可以实现对网络数据的实时采集，同时，对设备要求低，可以在原有低配置设备上直接进行改造，在很短的时间内完成所赋予的科研任务。

(上接第 114 页)

了多个对象，另外在声音数据的处理中分配了多块内存，这些都是系统资源，使用后必须释放掉，否则会造成系统资源流失和非法操作。

3 结束语

用 DirectSound 实现声音的实时捕获、处理与回放作为我们研制的基于 IP 网络的风洞试验指挥系统中实时语音通信所采用的主要技术，已在指挥系统得到了应用，能够实现实时语音通信，在网络中端到端的时延在 100ms 内，能满足指挥系统对实时性的要求。此技术也可以用于工程调度、电视电话会议等需要在 IP 网络中实时传输语音的场合。

参考文献：

[1] (美)Microsoft 公司. MSDN Library Visual Studio 6.0[M]. US:Microsoft,1998.

[2] (美)Microsoft 公司. Microsoft DirectX 7.0 SDK[M]. US: Microsoft,1999.

[3] (美)Richer J. Windows 核心编程[M]. 王建华，张焕生，侯丽坤，等译. 北京：机械工业出版社，2000.

[4] (美)Cadman C M. COM/DCOM 编程指南[M]. 刘 云，孔雷译. 北京：清华大学出版社，2000.