本文实例介绍了使用winmm.h进行音频流的获取的方法，具体步骤如下：

**一、首先需要包含以下引用对象**

|  |  |
| --- | --- |
| 1  2  3 | #include <Windows.h>  #include "mmsystem.h"  #pragma comment(lib, "winmm.lib") |

**二、音频的获取需要调用7个函数**

1. waveInGetNumDevs：返回系统中就绪的波形声音输入设备的数量

|  |  |
| --- | --- |
| 1 | UINT waveInGetNumDevs(VOID); |

2. waveInGetDevCaps：检查指定波形输入设备的特性

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | MMRESULT waveInGetDevCaps(   UINT\_PTR   uDeviceID,   LPWAVEINCAPS pwic,   UINT     cbwic  );  //uDeviceID 音频输入设备标识,也可以为一个打开的音频输入设备的句柄.  //  个人认为如果上一步获得了多个设备，可以用索引标识每一个设备.  //  //pwic 对WAVEINCAPS结构体的一个指针,包含设备的音频特性.  //  //cbwic WAVEINCAPS结构体的大小,使用sizeof即可.  //  //MMRESULT 函数执行的结果  //  MMSYSERR\_NOERROR 表示执行成功  //  MMSYSERR\_BADDEVICEID 索引越界  //  MMSYSERR\_NODRIVER 没有就绪的设备  //  MMSYSERR\_NOMEM 不能分配或者锁定内存 |

介绍WAVEINCAPS结构体的含义：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | typedef struct {    WORD   wMid;        //音频设备制造商定义的驱动程序标识    WORD   wPid;        //音频输入设备的产品标识    MMVERSION vDriverVersion;    //驱动程序版本号    TCHAR   szPname[MAXPNAMELEN];//制造商名称    DWORD   dwFormats;      //支持的格式,参见MSDN    WORD   wChannels;      //支持的声道数    WORD   wReserved1;      //保留参数  } WAVEINCAPS; |

3. waveInOpen：打开指定的音频输入设备，进行录音

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | MMRESULT waveInOpen(   LPHWAVEIN    phwi,        //接收打开的音频输入设备标识的HWAVEIN结构的指针   UINT\_PTR    uDeviceID,      //指定一个需要打开的设备标识.可以使用WAVE\_MAPPER选择一个按指定录音格式录音的设备   LPWAVEFORMATEX pwfx,        //一个所需的格式进行录音的WAVEFORMATEX结构的指针   DWORD\_PTR   dwCallback,    //指向一个回调函数、事件句柄、窗口句柄、线程标识,对录音事件进行处理.   DWORD\_PTR   dwCallbackInstance, //传给回调机制的参数   DWORD     fdwOpen      //打开设备的方法标识,指定回调的类型.参见CSDN  ); |

介绍WAVEFORMATEX结构体的含义：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | typedef struct {    WORD wFormatTag;    //波形声音的格式,单声道双声道使用WAVE\_FORMAT\_PCM.当包含在WAVEFORMATEXTENSIBLE结构中时,使用WAVE\_FORMAT\_EXTENSIBLE.    WORD nChannels;    //声道数量    DWORD nSamplesPerSec;  //采样率.wFormatTag为WAVE\_FORMAT\_PCM时,有8.0kHz,11.025kHz,22.05kHz,和44.1kHz.    DWORD nAvgBytesPerSec;  //每秒的采样字节数.通过nSamplesPerSec \* nChannels \* wBitsPerSample / 8计算    WORD nBlockAlign;    //每次采样的字节数.通过nChannels \* wBitsPerSample / 8计算    WORD wBitsPerSample;  //采样位数.wFormatTag为WAVE\_FORMAT\_PCM时,为8或者16    WORD cbSize;      //wFormatTag为WAVE\_FORMAT\_PCM时,忽略此参数  } WAVEFORMATEX; |

介绍dwCallback回调函数格式：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | void CALLBACK waveInProc(   HWAVEIN hwi,     //回调此函数的设备句柄   UINT uMsg,      //波形声音输入信息,标识关闭(WIM\_CLOSE)、缓冲区满(WIM\_DATA)、打开(WIM\_OPEN).   DWORD\_PTR dwInstance, //用户在waveInOpen指定的数据   DWORD\_PTR dwParam1,  //(LPWAVEHDR)dwParam1,用户指定的缓冲区   DWORD\_PTR dwParam2  ); |

4. waveInPrepareHeader：为音频输入设备准备一个缓冲区

|  |  |
| --- | --- |
| 1  2  3  4  5 | MMRESULT waveInPrepareHeader(   HWAVEIN hwi,  //音频输入设备句柄   LPWAVEHDR pwh,//指向WAVEHDR结构的指针,标识准备的缓冲区   UINT cbwh    //WAVEHDR结构的大小,使用sizeof即可  ); |

介绍WAVEHDR结构：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | typedef struct wavehdr\_tag {    LPSTR   lpData;     //指向波形格式的缓冲区    DWORD   dwBufferLength; //缓冲区的大小    DWORD   dwBytesRecorded; //当前存储了多少数据    DWORD\_PTR dwUser;     //用户数据    DWORD   dwFlags;      //为缓冲区提供的信息,在waveInPrepareHeader函数中使用WHDR\_PREPARED    DWORD   dwLoops;     //输出时使用,标识播放次数    struct wavehdr\_tag \* lpNext;//reserved    DWORD\_PTR reserved;     //reserved  } WAVEHDR, \*LPWAVEHDR; |

5. waveInAddBuffer：将缓冲区发送给设备，若缓冲区填满，则不起作用。（参数同上）

|  |  |
| --- | --- |
| 1  2  3  4  5 | MMRESULT waveInAddBuffer(   HWAVEIN hwi,   LPWAVEHDR pwh,   UINT cbwh  ); |

6. waveInStart：开始进行录制

|  |  |
| --- | --- |
| 1  2  3 | MMRESULT waveInStart(   HWAVEIN hwi //设备句柄  ); |

7. waveInClose：关闭设备

|  |  |
| --- | --- |
| 1  2  3 | MRESULT waveInClose(   HWAVEIN hwi //设备句柄  ); |

**三、完整实例代码如下:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110 | //Run.c文件  #include <Windows.h>  #include <stdio.h>  #include "mmsystem.h"  #pragma comment(lib, "winmm.lib")  void PlayMusi();  void WaveInitFormat(LPWAVEFORMATEX m\_WaveFormat, WORD nCh,DWORD nSampleRate,WORD BitsPerSample);  DWORD CALLBACK MicCallback(HWAVEIN hwavein, UINT uMsg, DWORD dwInstance, DWORD dwParam1, DWORD dwParam2);  void RecordWave();    void main()  {   //PlayMusi();   RecordWave();   while(1);  }  void RecordWave()  {   int count = waveInGetNumDevs();//1   printf("\n音频输入数量：%d\n",count);     WAVEINCAPS waveIncaps;   MMRESULT mmResult = waveInGetDevCaps(0,&waveIncaps,sizeof(WAVEINCAPS));//2   printf("\n音频输入设备：%s\n",waveIncaps.szPname);     if(MMSYSERR\_NOERROR==mmResult)   {   HWAVEIN phwi;   WAVEFORMATEX pwfx;   WaveInitFormat(&pwfx,1,8000,8);   printf("\n请求打开音频输入设备");   printf("\n采样参数：单声道 8kHz 8bit\n");   mmResult=waveInOpen(&phwi,WAVE\_MAPPER,&pwfx,(DWORD)(MicCallback),NULL,CALLBACK\_FUNCTION);//3     if(MMSYSERR\_NOERROR==mmResult)   {    WAVEHDR pwh1;    char buffer1[10240];    pwh1.lpData=buffer1;    pwh1.dwBufferLength=10240;    pwh1.dwUser=1;    pwh1.dwFlags=0;    mmResult=waveInPrepareHeader(phwi,&pwh1,sizeof(WAVEHDR));//4    printf("\n准备缓冲区1");      WAVEHDR pwh2;    char buffer2[10240];    pwh2.lpData=buffer2;    pwh2.dwBufferLength=10240;    pwh2.dwUser=2;    pwh2.dwFlags=0;    mmResult=waveInPrepareHeader(phwi,&pwh2,sizeof(WAVEHDR));//4    printf("\n准备缓冲区2\n");      if(MMSYSERR\_NOERROR==mmResult)    {    mmResult=waveInAddBuffer(phwi,&pwh1,sizeof(WAVEHDR));//5    printf("\n将缓冲区1加入音频输入设备");    mmResult=waveInAddBuffer(phwi,&pwh2,sizeof(WAVEHDR));//5    printf("\n将缓冲区2加入音频输入设备\n");      if(MMSYSERR\_NOERROR==mmResult)    {     mmResult=waveInStart(phwi);//6     printf("\n请求开始录音\n");    }    }   }     }  }  DWORD CALLBACK MicCallback(HWAVEIN hwavein, UINT uMsg, DWORD dwInstance, DWORD dwParam1, DWORD dwParam2)  {   switch(uMsg)   {   case WIM\_OPEN:    printf("\n设备已经打开...\n");    break;     case WIM\_DATA:    printf("\n缓冲区%d存满...\n",((LPWAVEHDR)dwParam1)->dwUser);    waveInAddBuffer (hwavein, (LPWAVEHDR)dwParam1, sizeof (WAVEHDR)) ;    break;     case WIM\_CLOSE:    printf("\n设备已经关闭...\n");    break;   default:    break;   }   return 0;  }  void WaveInitFormat(LPWAVEFORMATEX m\_WaveFormat, WORD nCh,DWORD nSampleRate,WORD BitsPerSample)  {   m\_WaveFormat->wFormatTag = WAVE\_FORMAT\_PCM;   m\_WaveFormat->nChannels = nCh;   m\_WaveFormat->nSamplesPerSec = nSampleRate;   m\_WaveFormat->nAvgBytesPerSec = nSampleRate \* nCh \* BitsPerSample/8;   m\_WaveFormat->nBlockAlign = m\_WaveFormat->nChannels \* BitsPerSample/8;   m\_WaveFormat->wBitsPerSample = BitsPerSample;   m\_WaveFormat->cbSize = 0;  }  void PlayMusi()  {   int error = mciSendString("open C:\\Users\\Angel\\Desktop\\有多少爱可以重来.mp3 alias myDivece", NULL, 0, NULL);   if (error == 0)   {   mciSendString("play myDivece", NULL, 0, NULL); //播放   }  } |