Encontré esto

```
$ apt-get install libvlccore-dev libvlc-dev
```

prueba.c:

```
#include <stdio.h>
#include <stdlib.h>

#include <vlc/vlc.h>

int main(int argc, char **argv)
{
    libvlc_instance_t *inst;
    libvlc_media_player_t *mp;
    libvlc_media_t *m;

    // load the vlc engine
    inst = libvlc_new(0, NULL);

    // create a new item
    m = libvlc_media_new_path(inst, "path to MP3 file");

    // create a media play playing environment
    mp = libvlc_media_player_new_from_media(m);

    // no need to keep the media now
    libvlc_media_release(m);

    // play the media_player
    libvlc_media_player_play(mp);
```

```
    sleep(10);

    // stop playing
    libvlc_media_player_stop(mp);

    // free the media_player
    libvlc_media_player_release(mp);

    libvlc_release(inst);


    return 0;
}
```

como enlazar y compilar:

```
$ gcc $(pkg-config --cflags libvlc) -c test.c -o test.o

$ gcc test.o -o test $(pkg-config --libs libvlc)
```

Después encontré esto

---

1. Primero de todo, escribir el siguiente código:

```
#include <Mmsystem.h>

#include <mciapi.h>

//these two headers are already included in the <Windows.h> header

#pragma comment(lib, "Winmm.lib")
```

2. Para abrir *.mp3:

```
mciSendString("open \"*.mp3\" type mpegvideo alias mp3", NULL, 0, NULL);
```

3. Jugar *.mp3:

```
mciSendString("play mp3", NULL, 0, NULL);
```

4. A play y esperar hasta que el *.mp3 ha terminado de jugar:

```
mciSendString("play mp3 wait", NULL, 0, NULL);
```

5. A replay (volver a jugar desde el inicio)*.mp3:

```
mciSendString("play mp3 from 0", NULL, 0, NULL);
```

6. Para repetir y esperar hasta que el *.mp3 ha terminado de jugar:

```
mciSendString("play mp3 from 0 wait", NULL, 0, NULL);
```

7. Para jugar el *.mp3 y reproducción cada vez que termina como un bucle:

```
mciSendString("play mp3 repeat", NULL, 0, NULL);
```

8. Si quieres hacer algo cuando el *.mp3 ha terminado de jugar, entonces usted necesita para `RegisterClassEx` por el `WNDCLASSEX` estructura, `CreateWindowEx` y proceso de los mensajes con el **GetMessage**, **TranslateMessage** y **DispatchMessage** funciones en un `while` bucle y llamada:

```
mciSendString("play mp3 notify", NULL, 0, hwnd);
```

9. En el procedimiento de ventana, añadir el `case MM_MCINOTIFY:` El código no se ejecutará cuando el mp3 ha terminado de jugar.

Pero si usted programa una **Consola** de la Aplicación y no tiene que lidiar con **windows**, entonces usted puede `CreateThread` en estado de suspensión especificando la `CREATE_SUSPENDED` bandera en el `dwCreationFlags` parámetro y mantener el valor de retorno en una `static` variable y llamarlo lo que quieras. Por ejemplo, yo me llamo mp3. El tipo de este `static` variable es `HANDLE` de curso.

Aquí es el `ThreadProc` para la `lpStartAddress` de este hilo:

```
DWORD WINAPI MP3Proc(_In_ LPVOID lpParameter) //lpParameter can be a pointer to a structure that store data that you cannot access outside of this function. You can prepare this structure before `CreateThread` and give it's address in the `lpParameter`

{

  Data *data = (Data*)lpParameter; //If you call this structure Data, but you can call it whatever you want.

  while (true)

  {

    mciSendString("play mp3 from 0 wait", NULL, 0, NULL);

    //Do here what you want to do when the mp3 playback is over
```

```
    SuspendThread(GetCurrentThread()); //or the handle of this thread that you keep in a static variable instead

  }

}
```

Todo lo que tienes que hacer ahora es `ResumeThread(mp3);` cada vez que se desea reproducir tus mp3 y algo va a suceder cada vez que se termina.

Puede `#define play_my_mp3 ResumeThread(mp3);` para hacer el código más legible.

Por supuesto, usted puede quitar el `while (true)`, `SuspendThread` y la **de 0** códigos, si quieres jugar con tu archivo mp3 sólo **una vez** y hacer lo que quiera cuando se termina.

Si **sólo** quitar el `SuspendThread` llamada, el sonido se reproducirá una y otra vez y hacer algo siempre es más. Esto es equivalente a:

```
mciSendString("play mp3 repeat notify", NULL, 0, hwnd); //or MAKELONG(hwnd, 0) instead
```

en windows.

Para pausar el *.mp3 de en medio:

```
mciSendString("pause mp3", NULL, 0, NULL);
```

y reanudar:

```
mciSendString("resume mp3", NULL, 0, NULL);
```

A parar en medio:

```
mciSendString("stop mp3", NULL, 0, NULL);
```

Tenga en cuenta que no se puede reanudar un sonido que ha sido detenido, pero sólo hizo una pausa, pero puede reproducir mediante la realización de la **jugar** comando.
Cuando termines de jugar este *.mp3, no te olvides de:

```
mciSendString("close mp3", NULL, 0, NULL);
```

Todas estas acciones también se aplican a (trabajar con) archivos de la onda, pero con la onda de los archivos, puede utilizar «waveaudio» en lugar de «mpegvideo».
También puedes jugar con ellos directamente, sin necesidad de abrirlos:

```
PlaySound("*.wav", GetModuleHandle(NULL), SND_FILENAME);
```

Si usted no desea especificar un identificador de módulo:

```
sndPlaySound("*.wav", SND_FILENAME);
```

Si usted no quiere esperar hasta que la reproducción es más:

```
PlaySound("*.wav", GetModuleHandle(NULL), SND_FILENAME | SND_ASYNC);
```

```
//or

sndPlaySound("*.wav", SND_FILENAME | SND_ASYNC);
```

Para reproducir el archivo de onda, una y otra vez:

```
PlaySound("*.wav", GetModuleHandle(NULL), SND_FILENAME | SND_ASYNC | SND_LOOP);

//or

sndPlaySound("*.wav", SND_FILENAME | SND_ASYNC | SND_LOOP);
```

Tenga en cuenta que debe especificar tanto el `SND_ASYNC` y `SND_LOOP` banderas, porque nunca se va a esperar hasta que un sonido, que se repite infinidad de veces, es lo más!

También puede `fopen` el archivo de onda y copiar todos sus bytes a un buffer (una enorme/enorme (muy grande) de la matriz de bytes) con el `fread` función y, a continuación:

```
PlaySound(buffer, GetModuleHandle(NULL), SND_MEMORY);

//or

PlaySound(buffer, GetModuleHandle(NULL), SND_MEMORY | SND_ASYNC);

//or

PlaySound(buffer, GetModuleHandle(NULL), SND_MEMORY | SND_ASYNC | SND_LOOP);

//or

sndPlaySound(buffer, SND_MEMORY);

//or

sndPlaySound(buffer, SND_MEMORY | SND_ASYNC);

//or

sndPlaySound(buffer, SND_MEMORY | SND_ASYNC | SND_LOOP);
```

Ya sea `OpenFile` o `CreateFile` o `CreateFile2` y bien `ReadFile` o `ReadFileEx` funciones se pueden utilizar en lugar de `fopen` y `fread` funciones.