Karlheinz Brandenburg / mp3

 A diez años de su gran invento. Por los años 96 y 97, Karlheinz Brandenburg, un científico alemán que trabajaba con su equipo en el instituto Fraunhofer terminaba de darle forma a sus investigaciones, jamás pensó que "eso" que estaba creando terminaría siendo una revolución en la forma de escuchar música a través de Internet, el MP3.
Y si bien este formato de audio digital comprimido ya lleva diez años en el mundo, el verdadero comienzo de la historia data de diez años antes, cuando en 1987 se iniciaron en el mismo instituto las investigaciones para crear una nueva forma de optimizar la transmisión de audio en formato digital.
Brandenburg, quien nació en 1954, en Earlangen (Alemania) y se graduó en la Universidad de dicha ciudad en Ingeniería electrónica en 1980 y dos años después, en Matemática, describió él mismo la forma en que se incorporó al proyecto: "A comienzos de los ochenta, en la época en que se digitalizó la red telefónica en Alemania, el profesor Seitzer, de la Universidad de Erlangen, tuvo la idea de transmitir por la línea telefónica RDSI de 64 kilobits por segundo algo más que la voz. Entonces comenzó a investigar un pequeño grupo en el que yo entré para hacer mi tesis de fin de carrera."

Su planteo era que para transmitir toda la información de un CD por la línea telefónica se necesitaba hacerlo a 1,5 megabits por segundo, pero sólo se podía transmitir 64 kilobits por segundo. Brandenburg pensó en que si se comprimía hasta doce veces la información, se podría transmitir por una sola línea telefónica.
Lo primero que desarrolló fue un codec, un programa que podía comprimir y descomprimir audios, manteniendo en gran parte la calidad del sonido.
Y de ahí al gran salto, al MP3 (las siglas o el diminutivo de MPEG-1 Audio Layer 3 (o MPEG-1 Capa de Audio 3).

"La posibilidad de intercambiar música por Internet la vimos desde el principio y se la ofrecimos a la industria... La respuesta que nos dieron entonces hoy les debe de

dar vergüenza: `Pero si nadie tiene Internet en casa´. Entonces ya teníamos la forma de codificarlo que permitía la venta por Internet, pero no supieron ver el potencial y nos mandaron a casa.”

El formato MP3 se convirtió en el estándar utilizado para streaming de audio y compresión de audio con pérdida de mediana fidelidad gracias a la posibilidad de ajustar la calidad de la compresión, proporcional a la tasa de bits (bitrate) y en consecuencia el tamaño final del archivo, permitiendo reducir hasta 12 e incluso 15 veces el del archivo original antes de su compresión.

Fue el primer formato de compresión de audio popularizado gracias a Internet, ya que hizo posible el intercambio de ficheros musicales. Los procesos judiciales contra empresas como Napster, AudioGalaxy y Megaupload son resultado de la facilidad con que se comparten legal e ilegalmente este tipo de ficheros, suponiendo el principal auge de la batalla por la propiedad intelectual en internet.

El formato MP3 se convirtió en el estándar utilizado para streaming de audio y compresión de audio con pérdida de mediana fidelidad gracias a la posibilidad de ajustar la calidad de la compresión, proporcional a la tasa de bits (bitrate) y en consecuencia el tamaño final del archivo, permitiendo reducir hasta 12 e incluso 15 veces el del archivo original antes de su compresión.
Fue el primer formato de compresión de audio popularizado gracias a Internet, ya que hizo posible el intercambio de ficheros musicales. Los procesos judiciales contra empresas como Napster, AudioGalaxy y Megaupload son resultado de la facilidad con que se comparten legal e ilegalmente este tipo de ficheros, suponiendo el principal auge de la batalla por la propiedad intelectual en internet.

Lenguaje C, formato MP3

Paquetes necesarios:

```
$ 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;
}
```

Enlazar y compilar:

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


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

Antes que nada, se escribe el siguiente codigo:

```
#include <Mmsystem.h>

#include <mciapi.h>

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

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

Para abrir *.mp3:

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

Reproducir *.mp3:

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

reproducir y esperar hasta que el *.mp3 se termine de reproducir:

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

Repetir (volver a reproducir desde el inicio) *.mp3:

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

Para repetir y esperar hasta que el *.mp3 se termine de reproducir:

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

Reproducción en bucle del *.mp3 (se vuelve a repetir al terminar la reproducción):

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

Si se quiere hacer algo cuando el *.mp3 ha terminado de reproducir, entonces se necesita: `RegisterClassEx` (registrar una clase) por el `WNDCLASSEX` estructura, `CreateWindowEx` y los mensajes con: **GetMessage**, **TranslateMessage** y **DispatchMessage** funciones en un `while` bucle y llamada:

```
mciSendString("play mp3 notify", NULL, 0, hwnd); //hwnd es un identificador del retorno de ventana por CreateWindowEx. Si esto no funciona, reemplazar el hwnd por MAKELONG(hwnd, 0).
```

En el procedimiento de ventana, añadir el `case MM_MCINOTIFY:` El código no se ejecutará cuando el mp3 se termine de reproducir.

Pero si se programa una **Consola** de la Aplicación y no se tiene que lidiar con **windows**, entonces se 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 como se desee. Por ejemplo, se llama mp3. El tipo la variable (`static`) 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 se tiene que hacer ahora es `ResumeThread(mp3);` cada vez que se desea reproducir tus mp3 y algo sucederá al terminar de reproducir.

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

De manera opcional se puede quitar el `while (true)`, `SuspendThread` y la **de 0** códigos, si se quiere reproducir con el 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);
```

se tiene que tener en cuenta que no se puede reanudar un sonido que ha sido detenido, pero sólo hizo una pausa, pero se puede reproducir mediante la realización del comando de **reproducción**.

Cuando este se termine de reproducir*.mp3:

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

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

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

Si no se especifica un identificador de módulo:

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

No esperar hasta que la reproducción 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);
```

Se debe especificar tanto el `SND_ASYNC` y `SND_LOOP` banderas, para no esperar a que se repita una infinidad de veces.

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.