Sonido en GNU/Linux: OSS, ALSA y OpenAL

Introducción

Hasta el momento, en esta sección hemos tratado el tema de la creación aplicaciones básadas en OpenGL, es decir, aplicaciones puramente gráficas. Durante el pasado tutorial se trató con especial importancia la portabilidad de esta librería, gracias a la cual las aplicaciones que la usan pueden compilarse en un gran número de arquitecturas y sistemas operativos.

En este artículo se va a realizar una introducción sobre como dotar de sonido a estas aplicaciones. En principio se presentarán dos metodos enfocados a GNU/Linux: OSS y ALSA, y en tercer lugar OpenAL, una magnifica librería que va a dotar a las aplicaciones que la usen de la misma portabilidad que tienen gracias a OpenAL. La razón de exponer los tres metodos y no hacerlo únicamente con OpenAL es que, hoy en dia, aun son pocas las aplicaciones que se han portado a esta nueva librería y muchas las que usan OSS.

¿Qué es el sonido?

Antes de comenzar a estudiar cómo programar aplicaciones es conveniente asegurar una pequeña base de conceptos relaccionados sobre que es el sonido.

El sonido es un fenómeno fisico que percibimos al producirse una perturbación en el medio en cual estamos; son ondas que se transmiten, por ejemplo, cuando tiramos una piedra al agua tranquila de un lago. A partir del lugar donde cayó la piedra, se generan ondas que se propagan en el agua. En el caso del sonido estas ondas se propagan por aire, sólidos y líquidos.

Los ordenadores en un comienzo no disponian de dispositivos sonoros, y de tenerlo se trataba de un pequeño altavoz (PC Speaker, por ejemplo). Actualmente cualquier PC casero dispone de una tarjeta de sonido, un periferico que transforma información digital y analogica, por lo tanto, en el caso de que el ordenador produzca un sonido se trata de una conversión digital-analogica (D/A), y en el caso de que lo "escuche" se trata de una conversión analogica-digital (A/D).

Driver de Linux y OSS

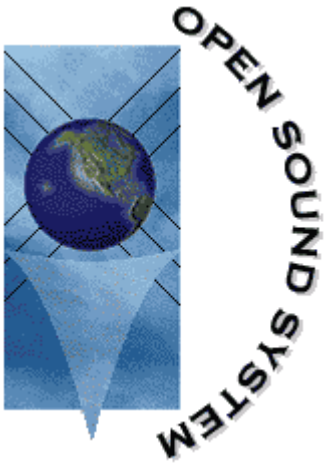
La primera de las opciones y más utilizada hoy en dia, son los drivers del kernel. Es por esta razón por la que se tratará esta opción en primer lugar.

OSS es una implementación propietaria de driver para tarjetas de sonido en máquinas UNIX. Para su desarrollo se basaron en los drivers desarrollados en el kernel de GNU/Linux. En la actualidad existen dos ramas de OSS, la propietaria y OSS/Free, una versión libre de OSS (licenciada bajo GNU GPL) que pasó a formar parte del kernel.

Aunque el futuro del soporte de sonido en GNU/Linux no se encuentra en OSS, si lo es el presente. Desde hace tiempo se viene desarrollando un proyecto para re-escribir y mejorar este sistema, aunque se va a mantener la compatibilidad con OSS. Más adelante trataremos este nuevo sistema: ALSA.

Cuando programamos una aplicación que hace uso del sistema de sonido, en primer lugar, debemos saber como acceder a él. Generalmente, existen tres puntos de acceso:

- **PCM:** El dispositivo de voz digital. Normalmente es el más utilizado. Sirve para producir y grabar sonido digital.

  Hay dos dispositivos en el directorio de devices relacionados: /dev/dsp y /dev/audio. Básicamente, tienen las mismas funcionalidades, aunque la información que manejan es diferente. /dev/dsp maneja información codificada en 8 bits sin signo, mientras que /dev/audio trabaja con información codificada con un algoritmo mu-law.

  Por otro lado, /dev/audio proporciona compatibilidad con el API de sonido de Sun Microsystems. Por lo general, las aplicaciones de GNU/Linux utilizan /dev/dsp y no /dev/audio.

  En una misma máquina es posible usar más de una tarjeta de sonido simultáneamente. Para dotar de este soporte a una máquina Linux, los distintos DSP se mapearán en los dispositivos /dev/dsp0, /dev/dsp1, /dev/dsp2, etc. Las aplicaciones que hagan uso de uno de estos dispositivos buscarán por defecto /dev/dsp, que será un link que apunte a la tarjeta de sonido que queremos utilizar por defecto. De esta forma, es posible configurar cualquier aplicación para que acceda a un DSP concreto y al mismo tiempo disponer de uno por defecto.

- **Mezclador:** Su uso básico consiste en manejar los niveles de volumen de las entradas y salidas (grabación y reproducción) de la tarjeta de sonido.

  A este dispositivo esta asociado una entrada en los devices de la máquina: /dev/mixer. En realidad, al igual que ocurria con /dev/dsp, este fichero es un link a uno de los siguientes dispositivos: /dev/mixer0, /dev/mixer1, etc. El fichero /dev/mixer? que sea enlazado por /dev/mixer será el mezclador por defecto.

- **Sintetizador:** Normalmente se usa para reproducir música sintetizada. Para aclarar el funcionamiento de este dispositivo se puede pensar en él como un interface MIDI.

A continuación se presenta un sencillo programa que realiza las inicializaciones necesarias para leer/escribir información, es decir sonido digital, en /dev/dsp. La finalidad de este ejemplo es que el lector se haga una idea de en que consiste este API, por razones de espacio no es posible tratarlo más en profundidad.

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/soundcard.h>

int
main (int argc, char **argv)
{
        int audio;
        int stereo     = 1;
        int tam_sample = 16;
        int veloci_dsp = 44100;
        int prof       = APF_NORMAL;

        int ret_error;

        audio = open ("/dev/dsp", O_RDONLY, 0);
        if (audio == -1) {
                        fprintf (stderr, "Imposible abrir /dev/dsp\n");
                        exit(1);
        }

        ret_error = tam_sample;
        ioctl(audio, SNDCTL_DSP_SAMPLESIZE, &tam_sample);
        if (tam_sample != ret_error) {
                fprintf (stderr, "Al fijar el tamaño de los samples\n");
                exit(1);
        }

        ioctl(audio, SNDCTL_DSP_PROFILE, &prof);

        ret_error = ioctl (audio, SNDCTL_DSP_STEREO, &stereo);
        if (ret_error == -1) {
                fprintf (stderr, "Error al fijar estereo\n");
                exit(1);
        }

        ret_error = ioctl (audio, SNDCTL_DSP_SPEED, &veloci_dsp);
        if (ret_error == -1) {
                fprintf (stderr, "Al fijar la velocidad del DSP\n");
                exit(1);
        }


        /*
         * En este punto ya es posible trabajar con el
         * descriptor 'audio' mediente las funciones de
         * sistema read() y write(), para por ejemplo,
         * emitir sonidos.
         *
         */

        close (audio);

        return 0;
}
```

Para más información sobre este API, se puede consultar un documento escrito por 4Front Technologies, la empresa que programa OSS, en él se puede encontrar toda la información necesaria. Al final del artículo se puede ver una referencia a este fichero.

ALSA

ALSA, siglas de Advanced Linux Sound Architecture, es un proyecto para mejorar la arquitectura de sonido de GNU/Linux. En la actualidad se encuentra en fase de desarrollo y aun no se ha integrado en la rama principal del kernel, aunque se espera que para la serie 2.5 de éste ya forme parte oficialmente.

Uno de los puntos fuertes de ALSA es que mantiene la compatibilidad con OSS, es decir, cualquier aplicación escrita para utilizar OSS podrá seguir ejecutándose aunque hayamos cambiado de arquitectura de sonido. Como es de esperar, ALSA dispone de su propio interface, además de la emulación de OSS, que en muchos aspectos mejora a este último. Actualmente son muchas las aplicaciones que han sido adaptadas a ALSA o que incorporan soporte para ambas arquitecturas.

El comienzo de este proyecto fue algo peculiar. En un principio, se trataba de escribir un driver para las tarjetas de sonido GUS (Gravis Ultra Sound). Una vez que el autor terminó con este propósito decidió comenzar a escribir ALSA, es decir, la arquitectura de sonido completa. El proyecto original de los driver de GUS aun se encuentra on-line con el nombre de "The Linux Ultra Sound Project" aunque no se actualiza desde 1998.

Al contrario que en el caso de OSS, ALSA dispone de una librería que funciona de interface con el API del Kernel. Por lo general, los programadores desarrollan aplicaciones que utilizan esta librería. Al proporciona todas las funcionalidades del kernel, ayuda a que la calidad del código de la aplicación sea superior, sin perjudicarla en nada. En la sección de referencias y direcciones de interes, al final de artículo, se puede encontrar la URL de la documentación completa de esta librería.

A continuación se presenta un sencillo ejemplo, que haciendo uso de la librería de ALSA, mira cuántas tarjetas de sonido hay instaladas en el sistema (siempre y cuando estén usando ésta arquitectura) y lee la información sobre cada una de ellas.

```
#include <stdio.h>
#include <string.h>
#include <sys/asoundlib.h>

int
main (int argc, char **argv)
{
     int idx;
        int cards;
        int err;
     snd_ctl_t *handle;
     struct snd_ctl_hw_info info;

     cards = snd_cards();
     printf("Detectada(s) %i tarjeta(s) de sonido.\n", cards);
```

```
    if (cards <= 0) {
                printf("Necesitas configurar ALSA primero\n");
            return 0;
    }

    for (idx = 0; idx < cards; idx++) {

             if ((err = snd_ctl_open(&handle, idx)) < 0) {
                   printf("Error al abrir: %s\n", snd_strerror(err));
                   continue;
             }

             if ((err = snd_ctl_hw_info(handle, &info)) < 0) {
                    printf("Error al coger información",
                           "sobre el hardware: %s\n",
                           snd_strerror(err));
                    continue;
             }

             snd_ctl_close(handle);
    }

    return 0;
}
```

Las funciones que comienzan por el prefijo "snd_" pertenecen a la librería de ALSA, es decir: snd_cards, snd_ctl_open, snd_strerror, snd_ctl_hw_info y snd_ctl_close son llamadas a la API de alto nivel de ALSA. En el paquete de distribución de ALSA se pude encontrar una detallada documentación con toda la información necesaria para utilizar esta librería.

OpenAL

OpenAL, Open Audio Library, es una librería con un gran futuro por delante. Su objetivo principal es el servir de capa de abstracción para el sistema de sonido de una aplicación. Gracias a este enfoque, la aplicación que haga uso de esta librería ganara muchisima portabilidad, por lo menos desde el punto de vista del soporte sonoro.

Podemos comparar OpenGL a OpenAL; una orientada a trabajar con gráficos de una forma transparente a la platarforma y la otra a hacerlo con sonido. Su enfoque a este respecto es el mismo que tiene OpenGL: trata de ser un estandar abierto y multiplataforma a la vez que neutral (no esta controlado por una compañia en concreto). Al igual que en el caso de OpenGL, OpenAL va a disponer de un ARB (Architecture Review and Acknowledgements), es decir un comité que controle su rumbo formado por distintas empresas interesadas.

El duo OpenGL + OpenAL es una combinación perfecta para desarrollar aplicaciones

portables con capacidades gráficas y sonoras, principalmente juegos o aplicaciones multimedia.

Otro de los puntos fuertes a destacar de OpenAL es su enfoque de trabajo con 'sonido 3D', gracias a lo cual, proporciona al programador la posibilidad de reproducir sonidos en una aplicación que simulen el sonido real de un espacio 3D. Básicamente, para la reproducción de sonido con OpenAL es necesario especificar en un espacio tridimensional, un punto de emisión y un punto en el que se encuentra el receptor que lo va a escuchar, despues de lo cual el sonido elegido habrá sido procesado y colocado en un buffer de salida.

Programación con OpenAL

En este apartado trataremos la programación de OpenAL desde un punto de vista parcialmente practico. El API de OpenAL es demasiado extenso para poderlo tratar en este artículo. De cualquier forma, para profundizar más en la librería siempre es posible consultar los ficheros de cabeceras (.h). En la implentación de Loki Software la mayoria de las funciones se encuentran muy bien documentadas. En una máquina Linux, normalmente son accesibles en:

```
/usr/include/AL/al.h
/usr/include/AL/altypes.h
```

En OpenAL existen tres tipos fundamentales de datos:

- Fuentes (sources): Estos objetos contienen y manejan la información referente al espacio 3D. Contienen información sobre la situación y la dirección del sonido. Cada una de las fuentes de sonido va a tener asociado un buffer.
- Buffers: Estos objetos son referenciados por las fuentes. Normalmente son menores en número que estas ultimas. Su función consiste en almacenar sonido, ya sea comprimido y descomprimido.
- Listerner: Existe un único listener por cada contexto de sonido de la aplicación. Esta clase de objeto contiene propiedades parecidas a las de una fuente que se tendrán en cuenta a la hora de realizar el procesado del sonido.

Para proporcionar un nivel de portabilidad alto, OpenAL tiene en cuenta tambien los tipo de dato básicos, tales como números enteros, números de coma flotante o valores booleanos. Para ello dispone, al igual que OpenGL, de un conjunto de tipos propios. Estos tipos, tienen de la peculiaridad de coincidir con los de OpenGL cambiando únicamente el prefijo de estos. En OpenGL estos tipos reciben el prefijo de "GL" mientras que en OpenAL es "AL". Por ejemplo un valor booleano en OpenGL tiene un tipo de "GLboolean", mientras que en OpenAL es "ALboolean".

Las aplicaciones que utilizan OpenAL suelen seguir unos pasos generales en cuanto al manejo de sonido se refiere. Por ejemplo, la siguiente lista recoge los pasos típicos de una pequeño programa básado en OpenAL:

- Crear un contexto

- Crear un conjunto de buffers
- Carga de sonido en los buffers
- Crear las fuentes (sources)
- Referenciar fuentes a buffers
- Fijar localización y direcciones
- Fijar los valores del estado inicial de OpenAL

A continuación se presentará un programa que, haciendo uso de OpenAL, carga y reproduce un fichero WAV. A través de este ejemplo podemos ver cada uno de los pasos explicados anteriormente.

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

#include <AL/al.h>
#include <AL/alc.h>
#include <AL/alext.h>

int
main (int argc, char **argv)
{
       ALCdevice *dev;
       void      *id_contexto = NULL;
       void      *wave        = NULL;

       ALfloat ceros[]    = {0.0f, 0.0f,  0.0f};
       ALfloat atras[]    = {0.0f, 0.0f, -1.0f, 0.0f, 1.0f, 0.0f};
       ALfloat position[] = {2.0f, 0.0f, -4.0f};

       ALuint    boom;
       ALuint    multis;
       ALsizei   tamano;
       ALsizei   bits;
       ALsizei   frec;
       ALsizei   formato;
       ALboolean err;
       ALint     estado;


       dev = alcOpenDevice (NULL);
       if (dev == NULL) {
                 fprintf (stderr, "Error al abrir dispositivo\n");
               return 1;
       }

       id_contexto = alcCreateContext (dev, NULL);
       if (id_contexto == NULL) {
                 fprintf (stderr, "Error al crear el contexto\n");
                       alcCloseDevice (dev);
                return 1;
        }

        alcMakeContextCurrent (id_contexto);

        alListenerfv (AL_POSITION, ceros);
        alListenerfv (AL_VELOCITY, ceros);
```

```
        alGenBuffers (1, &boom);

        err = alutLoadWAV ("ejemplo.wav", &wave, &formato,
                           &tamano, &bits, &frec);
        if(err == AL_FALSE) {
               fprintf (stderr, "Imposible abrir el ejemplo.wav\n");
               exit(1);
        }

        alBufferData (boom, formato, wave, tamano, frec);
        free (wave);


        alGenSources (1, &multis);

        alSourcefv (multis, AL_POSITION, position);
        alSourcefv (multis, AL_VELOCITY, ceros);
        alSourcefv (multis, AL_ORIENTATION, atras);

        alSourcei (multis, AL_LOOPING, AL_FALSE);
        alSourcef (multis, AL_GAIN_LINEAR_LOKI, 1.0);

        alQueuei (multis, AL_BUFFER, boom);

        alSourcePlay (multis);

        alGetSourceiv(multis, AL_SOURCE_STATE, &estado);
        while (estado == AL_PLAYING) {
                usleep (1000);
                alGetSourceiv(multis, AL_SOURCE_STATE, &estado);
        };

        alcMakeContextCurrent (NULL);
        alcDestroyContext (id_contexto);

        alcCloseDevice (dev);

        return 0;
}
```

Para compilar este pequeño ejemplo, basta con tener OpenAL instalado en el sistema y ejecutar:

```
gcc ejemplo_openal.c -o ejemplo_openal -lopenal
```

Conclusiones

A lo largo del artículo se han presentado las tres formas principales, que no las únicas, de escribir aplicaciones con soporte sonoro en GNU/Linux, cada una de las cuales con una ventajas evidentes:

- OSS/Free: Es el sistema mayoritario hoy en día, cuenta con una gran difusión y es muy sencillo encontrar información y código fuente sobre él.

- ALSA: Es la arquitectura de sonido que se va a implantar en GNU/Linux dentro de poco; una apuesta segura.
- OpenAL: Tiene como principal ventaja la portabilidad, ser un estandar abierto y su enfoque al desarrollo de juegos y aplicaciones multimedia.

Referencias

- OSS/Free http://www.linux.org.uk/OSS
- OSS http://www.opensound.com
- Documentación OSS http://www.opensound.com/pguide/oss.pdf
- ALSA http://www.alsa-project.org
- The Linux Ultra Sound Project http://www.perex.cz/~perex/ultra
- Documentación de ALSA-lib http://www.alsa-project.org/alsa-doc/alsa-lib
- Repositorio de OpenAL http://cvs.lokigames.com/cgi-bin/cvsweb.cgi/openal
- Loki Games http://www.lokigames.com