Compilación cruzada

Créditos

» **Autor:** Hugo Ruscitti
» **Fecha:** 31 de Agosto del 2006

Introducción

Este trabajo presenta una serie de consejos e indicaciones sobre como generar programas multimedia para Windows desde un sistema operativo como GNU/Linux.

Dado que la temática que trataremos es de cierta complejidad, he decidido no crear una referencia completa sobre las herramientas que utilizaremos, en su lugar intentaré abordar el asunto de manera práctica, creando ejemplos de aplicaciones funcionales. De esta forma podrá conocer los componentes mas elementales del proceso y ver sus resultados inmediatos.

El por qué de la cuestión

GNU/Linux cuenta con una extensa galería de recursos para quienes disfrutamos de la programación. Entre aplicaciones y bibliotecas, GNU/Linux ofrece un entorno confortable y flexible, incluso, para desarrolladores de aplicaciones multimedia.

Una herramienta para programadores muy interesante es Mingw ("Minimalistic GNU for Windows"), que nació con la intensión de adaptar varias herramientas de desarrollo del sistema GNU a entornos Windows.

Si bien encontrará varios usos para Mingw, aquí lo utilizaremos para lograr que nuestro sistema GNU pueda generar archivos ejecutables para sistemas Windows. En general, a este proceso de "exportar" aplicaciones a otros sistemas o arquitecturas se lo denomina "cross-building", "cross-compile", o como decidimos llamarlo en este artículo: "compilación cruzada".

Este recurso de soporte a otros sistemas nos permite ampliar nuestras posibilidades de crear programas Multiplataforma, sin cambiar de sistema y utilizando siempre Software Libre.

Requisitos

Para seguir este tutorial le recomiendo contar con algunas herramientas que veremos mas adelante: gcc, mingw y wine. Muchas distribuciones de GNU/Linux cuentan con sistemas administradores de paquetes que le permiten instalar y desinstalar programas con facilidad, por ese motivo no debería tener problemas en comenzar a probar los ejemplos de este artículo.

El proceso de instalación en mi sistema Debian GNU/Linux llevó unos pocos segundos. Si utiliza este mismo sistema, u otro basado en él como Ubuntu o Knoppix, ejecute como usuario administrador (root) 1:

```
apt-get install mingw32
```

El sistema de paquetes `apt` le informará que necesita instalar, como mínimo, otros 2 paquetes llamados `mingw32-runtime` y `mingw32-binutils`. Indique Sí y continúe.

Si todo sale bien obtendrá un mensaje como:

```
Configurando mingw32-binutils (2.15.94-20050118.1-1) ...
Configurando mingw32-runtime (3.7-1) ...
Configurando mingw32 (3.4.2.20040916.1-2) ...
```

En cambio, si se le presenta algún tipo de problema puede recurrir a las versiones en código fuente disponibles en el sitio oficial de Mingw, o bien, obtener un entorno de compilación completo en el sitio de la biblioteca SDL.

Nuestro programa ejecutable para GNU/Linux

Seguramente alguna vez ha escrito el clásico programa que imprime en pantalla la leyenda "hola mundo !". Este sencillo programa nos resultará de mucha utilidad como primer ejemplo.

```
#include <stdio.h>

int main (void)
{
 printf ("Hola mundo!\n");
 printf ("\n");
 printf ("pulse ENTER para continuar\n");

 getchar ();

 return 0;
}
```

Para crear un programa ejecutable compatible con GNU/Linux podemos invocar al compilador gcc de la siguiente manera:

```
gcc hola_mundo.c
```

Mediante dicho comando, se generará un archivo ejecutable llamado `a.out`. Ahora estamos en condiciones de ejecutar el programa escribiendo:

```
./a.out
```

Tenga en cuenta que también podría alterar el nombre del programa generado (a.out) especificando el comando `-o` como se muestra a continuación:

```
gcc hola_mundo.c -o hola
```

de forma que el archivo generado reciba el nombre `hola` en lugar de `a.out`.

Compilando para sistemas Windows con MinGW

El programa anterior funcionaría solamente en GNU/Linux. Si nuestra intensión es generar una versión funcional para sistemas Windows podemos recurrir a MinGW. La forma de utilizar esta herramienta en GNU es muy similar al resto de los programas como gcc y make, en Mingw encontrará una serie de programas que cumplen las tareas de `procesar`, `compilar` y `enlazar` aplicaciones pero generando archivos ejecutables para sistemas Windows.

En nuestro caso, el programa que reemplaza la tarea de generar el programa ejecutable será i586-mingw32-gcc, por lo tanto si ejecutamos:

```
i586-mingw32msvc-gcc hola_mundo.c -o hola.exe
```

Obtendremos el archivo ejecutable hola.exe para sistemas Windows. Note que he utilizado la opción `-o` como en el ejemplo de comando anterior para definir el nombre del programa.

gcc no es único programa incluido en Mingw, existen muchos otros bajo la misma nomenclatura, donde se utiliza el prefijo i586-mingw32msvc. Por ejemplo, para generar programas en lenguaje C++ puede utilizar el programa i586-mingw32msvc-g++.

Analizando el resultado de la compilación

El programa que generamos con MinGW ya está en condiciones de funcionar en un sistema operativo como Windows, solo debería trasladar el programa ejecutable a ese sistema junto con su biblioteca mingwm10.dll. Aunque también existe la posibilidad de analizar el funcionamiento del programa desde nuestro propio sistema GNU/Linux, simulando un entorno de funcionamiento Windows.

Existe un programa muy popular en GNU que nos permite interpretar aplicaciones para Windows, este programa se llama Wine, y se incluye en numerosos repertorios de programas. En el caso de las distribuciones basadas en Debian GNU/Linux es probable que necesite instalarlo mediante el sistema `apt`. Para ello ejecute como administrador del sistema (root) el siguiente comando:

```
apt-get install wine winesetuptk
```

El segundo programa (winesetuptk) se utiliza para definir los parámetros de configuración de wine de manera sencilla.

Cuando termine de instalar el programa podrá correr aplicaciones diseñadas para sistemas Windows desde su sistema GNU invocando a los comandos:

- **»** wine: para cargar y ejecutar programas gráficos.
- **»** wineconsole: para cargar y ejecutar programas de consola.

Para verificar que nuestro programa `hola.exe` funciona mediante wine podemos ejecutar:

```
wineconsole hola.exe
```

Note que los programas para Windows suelen clasificarse en dos grupos: los programas de `consola` y las aplicaciones `gráficas`. Aquí comencé con un programa sencillo para mostrar que puede compilarlo tanto para

GNU como Windows, este primer ejemplo es un programa de `consola`.

Analicemos otro ejemplo; en la documentación del programa MinGW existe un programa que utiliza rutinas propias de los sistemas Windows. Lo utilizaremos para mostrar como funciona Wine con un programa gráfico.

```
#include <windows.h>

int main(int argc, char *argv[])
{
 MessageBox(NULL, "Hello, world!", "Hello, world!", MB_OK);
 return 0;
}
```

Si genera un archivo ejecutable en base a este programa gráfico mediante Mingw32, luego podrá invocar al programa wine como se muestra a continuación:

```
wine hello.exe
```

Obteniendo en pantalla:

Si bien este no es un programa de `consola` debería tener en cuenta que en los sistemas Windows aparecerá, de todas maneras, una pantalla de `consola`. Para evitar que esto ocurra debe indicarle a MinGW que el programa será completamente `gráfico` mediante la opción `-mwindows`:

```
-mwindows
```

Utilizando bibliotecas

MinGW también puede establecer vínculos entre nuestros programas y diversas bibliotecas dinámicas, simplemente debe obtener las bibliotecas de desarrollo que necesite, colocarlas en el directorio adecuado e indicarle a MinGW que las utilizará.

Para mostrar esta posibilidad tomaremos como ejemplo un programa multimedia que utiliza la biblioteca SDL llamado Mouse. Primero instalaremos el entorno de desarrollo, luego generamos el programa y por último analizaremos su funcionamiento mediante `wine`.

Instalando la biblioteca SDL

En el sitio web www.libsdl.org encontrará una versión de la biblioteca indicada como `mingw32`, al momento de

escribir este artículo la versión mas reciente es 1.2.11 y el archivo para descargar recibe el nombre de sdl-devel-1.2.11-mingw32.tar.gz. De todas formas es recomendable que visite la web de SDL y busque si existe una versión mas reciente.

Para instalar esta biblioteca he tenido que ejecutar:

```
tar xzvf sdl-devel-1.2.10-mingw32.tar.gz
cd SDL-1.2.10
su
mkdir /usr/local/cross-tools/
mkdir /usr/local/cross-tools/i386-mingw32msvc/
make cross
make install
exit
```

Si por algún motivo quiere alterar el directorio de instalación de la biblioteca deberá editar el archivo Makefile incluido en el directorio SDL-1.2.10 y reemplazar la linea:

```
CROSS_PATH := /usr/local/cross-tools/i386-mingw32msvc/
```

Compilar un programa que utiliza SDL para GNU

Antes de continuar recordemos como utilizar la utilidad `sdl-config` para crear aplicaciones nativas en GNU/Linux:

El proceso de compilación de un programa que utiliza bibliotecas requiere de varias especificaciones (que bibliotecas enlazar, donde residen las cabeceras .h, etc). En GNU/Linux estas especificaciones se suelen indicar mediante programas (o scripts) que obtienen todos los parámetros necesarios para realizar la compilación, en el caso de las bibliotecas SDL existe el comando sdl-config, que nos facilita la tarea de realizar especificaciones. Por ejemplo:

```
sdl-config --cflags
```

nos informa todos los parámetros necesarios para iniciar el proceso de compilación (generar los ficheros objeto). Además:

```
sdl-config --libs
```

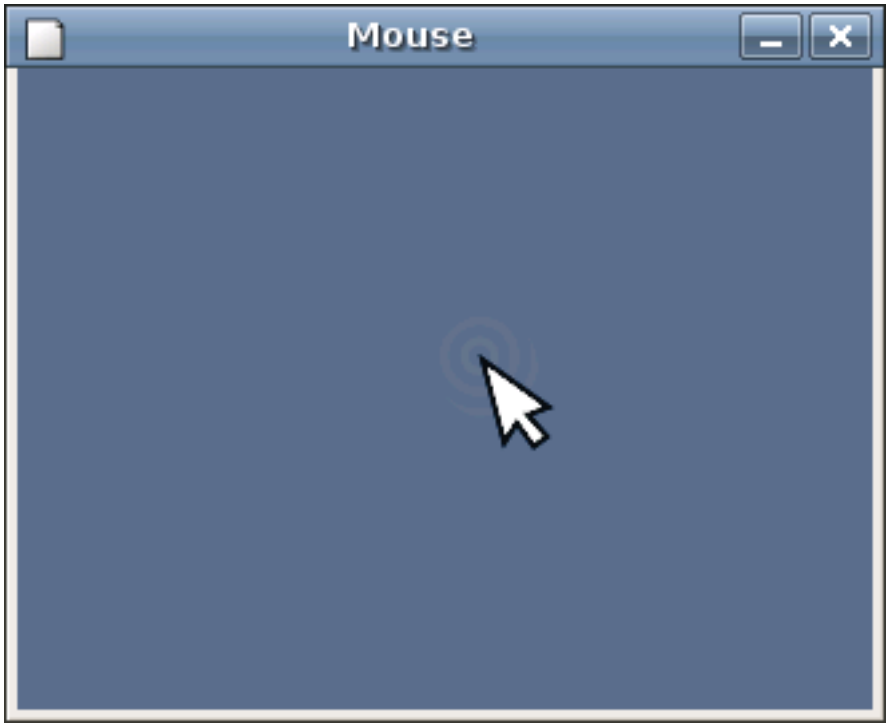
nos indica las opciones que debemos especificar para enlazar el programa a las bibliotecas. Así, nuestro programa se puede compilar en GNU/Linux mediante el comando:

```
gcc mouse `sdl-config --cflags --libs`
```

Las comillas invertidas (``) le permiten a gcc obtener el retorno de sdl-config y utilizarlo como argumento. En este caso sería equivalente invocar a sdl-config --cflags --libs, copiar su resultado en pantalla y colocarlo como argumento al generar el programa con gcc.

Una vez invocada la compilación, nuestro programa de ejemplo se podrá ejecutar mediante el comando:

```
./mouse
```



Nuestro programa funcionando en GNU/Linux

Compilar un programa con SDL para Windows desde GNU

El proceso de compilación mediante MinGW es muy similar, solo que debemos reemplazar la llamada a sdl-config por un script situado en:

```
/usr/local/cross-tools/i386-mingw32msvc/bin/sdl-config
```

Este script funciona de manera similar a la herramienta sdl-config nativa de GNU/Linux. Por ejemplo si queremos obtener todas las opciones de compilación y enlace para nuestros programas con gcc invocamos a:

```
/usr/local/cross-tools/i386-mingw32msvc/bin/sdl-config --cflags --libs
```

Para probar nuestro ejemplo completo, ejecutemos las siguientes sentencias en un intérprete de órdenes:

```
alias gcc_cross=i586-mingw32msvc-gcc
alias sdl-config_cross=/usr/local/cross-tools/i386-mingw32msvc/bin/sdl-config

gcc_cross mouse.c -o mouse.exe `sdl-config_cross --cflags --libs`
```

Note que las llamadas a `alias` nos facilitan escribir los comandos completos. Para analizar otra alternativa de compilación puede descargar el ejemplo completo e invocar al programa Make.

En caso de errores ...

Si el proceso de compilación falla indicando que no se encuentran las cabeceras SDL.h, posiblemente se deba a un error del script `sdl-config`. Note que las bibliotecas SDL se suelen instalar dentro de un directorio llamado "include" o "include/SDL", el script `sdl-config` debe indicar con precisión el lugar exacto en donde residen estos archivos.

Lamentablemente he notado que la versión actual de sdl-config no indica correctamente la ruta de estos archivos .h en mi sistema. Si en su equipo ocurre lo mismo intente editar el archivo /usr/local/cross-tools/i386-mingw32msvc/bin/sdl-config como administrador de sistema (root) y cambie la linea:

```
echo -I${prefix}/include/SDL -D_GNU_SOURCE=1 -Dmain=SDL_main
```

por:

```
echo -I${prefix}/include -D_GNU_SOURCE=1 -Dmain=SDL_main
```

Si el problema persiste no dude en consultarlo en nuestro foro de mensajes.

Otro problema muy frecuente en la utilización de esta biblioteca se encuentra en la inclusión del archivo "SDL.h" al código de nuestro programa. La forma mas adecuada de incluir este archivo es mediante:

```
#include "SDL.h"
```

Es cierto que existen muchas formas de indicar la ruta a un archivo como este ("SDL/SDL.h", "sdl.H" etc.), aunque sería deseable que adopte solo aquella forma que le brinda mayores posibilidades de generar el programa en diferentes sistemas sin cambiar una sola linea del programa.

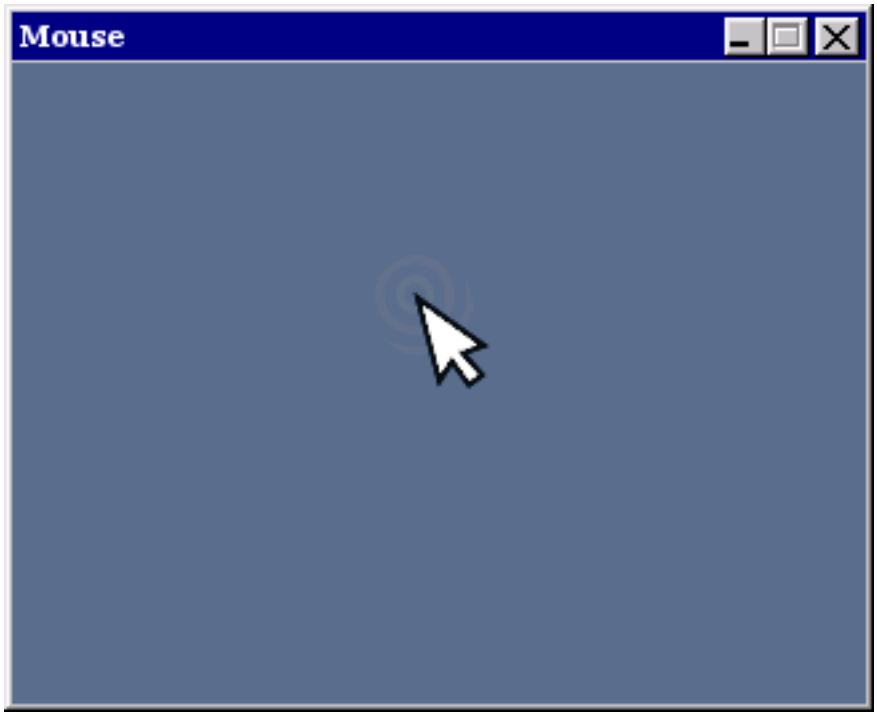
Verificando el funcionamiento del programa:

A diferencia de nuestro primer ejemplo, este programa se encuentra vinculado a la biblioteca dinámica SDL y por ese motivo necesitará del archivo SDL.dll para que todo funcione correctamente. Tanto en sistemas Windows como mediante Wine podrá resolver esta dependencia situando el archivo SDL.dll dentro del directorio donde reside el archivo ejecutable:

```
cp /usr/local/cross-tools/i386-mingw32msvc/bin/SDL.dll ./
```

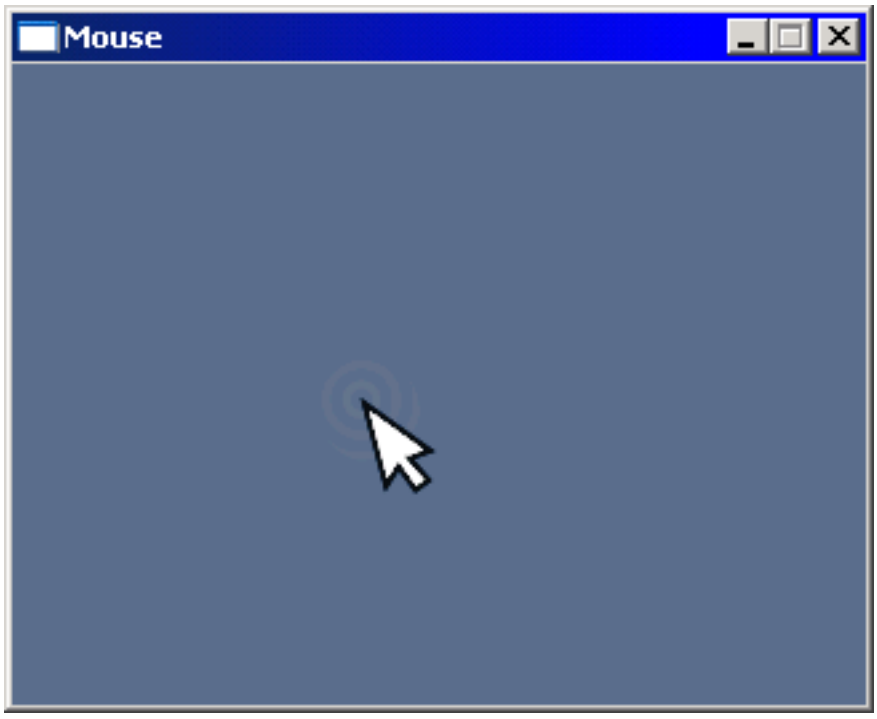
Ahora observemos como funciona nuestro programa con Wine:

```
wine mouse.exe
```



Programa mouse sobre Wine en Debian GNU/Linux

si ejecutamos este mismo programa en un sistema Windows notaremos que su funcionalidad es idéntica:



Funcionando sobre Windows XP con el tema de escritorio clásico

Si utiliza programas SDL con frecuencia, tal vez le resulte mas útil situar el archivo SDL.dll dentro de la carpeta "Windows\system" (en el sistema Windows) o en "$HOME/.wine/fake_windows/Windows/System" (si utiliza Wine). Esto evitará que necesite copiar el archivo .dll en cada proyecto que realice.

Bibliotecas Adicionales

SDL cuenta con una serie de bibliotecas adicionales para brindar soporte a otras tareas como utilizar ficheros SVG (formato de imágenes vectoriales), MP3 (música), GUI (interfases visuales), etc. Añadir estas bibliotecas a su proyecto es muy sencillo, solo debe obtener las bibliotecas de desarrollo que necesite para sistemas Windows (o aquellas indicadas para mingw32) y ubicar los archivos .h dentro del directorio:

```
/usr/local/cross-tools/i386-mingw32msvc/include/
```

Luego necesitará copiar los archivos de biblioteca (.a/.la) en:

```
/usr/local/cross-tools/i386-mingw32msvc/lib/
```

Por último, no olvide que los usuarios de sus aplicaciones necesitarán tener los archivos .dll en su sistema a la hora de ejecutar los programas.

Conclusión

GNU/Linux incluye una extensa colección de programas interesantes para todo tipo de usuarios. A mi entender, Mingw es una herramienta indispensable para aquellos programadores que estudiamos el desarrollo de programas libres y Multiplataforma. Aquí solo hemos visto el principio de la historia ...

Licencia

Notas

1 - En distribuciones como Ubuntu puede ejecutar `sudo apt-get ...` como usuario normal.

Este documento ha sido generado automáticamanete a partir del archivo 'compilacion_cruzada.xml' el Mon Jan 19 20:32:41 2009
La versión mas reciente de este documento se almacena en www.losersjuegos.com.ar. Visitenos para obtener mas recursos y actualizaciones.