Instalación de Ginga

La información para la instalación del software emulador de Ginga fue obtenido de distintas páginas dedicadas al desarrollo de material interactivo para televisión digital. Al final de cada sección de documentación se especifica las fuentes y las fechas de consulta.

La instalación fue realizada desde los fuentes. La versión que se utilizó fue la actualizada al 27 de mayo de 2011. Versión estable 1.1.0

Instalación de las dependencias

Para la distribución utilizada en la máquina de trabajo (Ubuntu 10.04 LTS) se utilizó la siguiente entrada para descargar e instalar las dependencias:

```
sudo apt-get install libtool autoconf automake g++ libssl-dev pkg-config libpng12-dev libdirectfb-dev cmake libavformat-dev libxine-dev liblua5.1-0-dev liblua5.1-socket2 libcurl4-openssl-dev libxerces-c-dev libboost-dev libdb4.7-dev libboost-thread-dev libboost-filesystem-dev libboost-regex1.40-dev libavcodec-dev libxine1-ffmpeg libgtest-dev
```

Instalación del Fusion Sound
El Fusion Sound será compilado desde las fuentes, ya que muy pocas distribuciones de Linux tienen este incluido. Se instalará la que se encuentra en el repositorio GIT con el siguiente comando:

```
git clone git://git.directfb.org/git/directfb/core/FusionSound.git
```

En el caso personal, no se tenía instalado el GIT. Para descargar e instalarlo se necesita usar el comando siguiente:

```
apt-get install git
```

Una vez instalado el git, se puede reintentar la instalación del Fusion Sound como se encuentra especificado anteriormente.

Ahora que se tienen descargadas las fuentes, se las debe compilar. Usualmente se utiliza el comando **./configure.** En este caso en particular utilizaremos el comando **./autogen.sh.**
Es importante mencionar que para la compilación se deben tener las herramientas adecuadas actualizadas. Aquí se presenta una lista de las herramientas que se deberían actualizar antes de hacer la compilación:

- **libtool** (Generic Library Support Script) (versión mayor o igual a 1.3.4)
- **autoconf** (Generador automático de guiones de configuración) (versión mayor o igual a 2.13)
- **automake** (Herramienta para crear archivos Makefile que cumpla con archivos GNU)(versión mayor o igual a 1.4)

Una vez que hayamos instalado y verificado las actualizaciones de las herramientas de la distribución entramos a la carpeta del FusionSound que se creó al momento de la instalación y procedemos con la compilación con el siguiente código:

```
cd FusionSound
./autogen.sh
```

Después de ejecutar el comando anterior se verá una lista de verificaciones de software actualizado necesario. Se debe prestar mucha atención a esta lista, pues en el caso que se utilice una instalación de Ubuntu reciente es probable que muchas herramientas y librerías estén desactualizadas. Actualice todas aquellas que presentaron error. Posteriormente introduzca el siguiente comando para compilar.

```
make
```

```
sudo make install
```

Una vez ejecutado este código, habremos compilado el FusionSound.

Obtención de las fuentes Ginga

Ahora que se tienen todas las dependencias instaladas se puede proceder con la compilación de Ginga. Es importante verificar que todos los anteriores pasos hayan sido ejecutados con éxito, pues en el caso contrario, la compilación de Ginga fracasará.

Antes de la descarga de las fuentes, nos debemos asegurar que tenemos instalada la herramienta para poder descargar con el comando “svn” de subversion. En el caso que no se tenga esta herramienta, se la puede instalar con el siguiente comando:

apt-get install subversion
svn checkout svn://scm.ourproject.org/svnroot/ginga/middleware/ginga.ar/tags/release/1.1.0

Este proceso puede durar un par de minutos, pues necesita descargar todos los archivos necesarios para la compilación de la parte más importante del emulador de Ginga NCL.

Aplicación del parche Ginga.ar v1.1.0

Después de haber descargado las fuentes satisfactoriamente, se debe instalar un parche que corrige algunos errores de la version 1.1.0 con las nuevas versiones del FusionSound. Es importante mencionar que la dirección en la que se ejecuta el comando de patch debe ser la carpeta /1.1.0. De otra forma no funcionará.

El archivo de parche debe tener el nombre: “ginga.fusionsound.patch”. Y debe tener el siguiente contenido:

```
Index: src/gingacc-system/src/io/interface/content/audio/FFmpegAudioProvider.cpp
===================================================================
--- src/gingacc-system/src/io/interface/content/audio/FFmpegAudioProvider.cpp   (revision 9)
+++ src/gingacc-system/src/io/interface/content/audio/FFmpegAudioProvider.cpp   (working copy)
@@ -829,11 +829,12 @@
                         dsc.sampleformat = FSSF_S16;

                         rContainer->audio.sound = FusionSoundAudioProvider::_fsSound;
-                        retB = rContainer->audio.sound->CreateStream(
+                        // FIXME: La funciÃ³n CreateStream devuelve DirectResult!
+                        ret = rContainer->audio.sound->CreateStream(
                                         rContainer->audio.sound,
                                         &dsc, &rContainer->audio.stream);

-                        if (retB != DFB_OK) {
+                        if (ret != DR_OK) {
                                 cout << "FFmpegAudioProvider::initializeFFmpeg ";
                                 cout << "IFusionSound::CreateStream() failed!";
                                 cout << endl;
@@ -1165,16 +1166,19 @@

         void FFmpegAudioProvider::setSoundLevel(float level) {
                 DFBResult retB;
+                DirectResult ret;

                 if (level < 0.0) {
                         return;
                 }
```

```
                if (rContainer->audio.playback) {
-                       retB = rContainer->audio.playback->SetVolume(
+
+                       // FIXME: La funciÃ³n SetVolume devuelve DirectResult!
+                       ret = rContainer->audio.playback->SetVolume(
                                        rContainer->audio.playback, level);

-                       if (retB == DFB_OK) {
+                       if (ret == DR_OK) {
                                rContainer->volume = level;
                        }
                }
Index: src/gingacc-system/src/io/interface/content/video/FFmpegVideoProvider.cpp
===================================================================
--- src/gingacc-system/src/io/interface/content/video/FFmpegVideoProvider.cpp   (revision 9)
+++ src/gingacc-system/src/io/interface/content/video/FFmpegVideoProvider.cpp   (working copy)
@@ -1071,11 +1071,12 @@
                        dsc.sampleformat = FSSF_S16;

                        rContainer->audio.sound = FusionSoundAudioProvider::_fsSound;
-                       retB = rContainer->audio.sound->CreateStream(
+                       // FIXME: DirectResult en vez de DFBResult
+                       ret = rContainer->audio.sound->CreateStream(
                                        rContainer->audio.sound,
                                        &dsc, &rContainer->audio.stream);

-                       if (retB != DFB_OK) {
+                       if (ret != DR_OK) {
                                cout << "FFmpegVideoProvider::initializeFFmpeg ";
                                cout << "IFusionSound::CreateStream() failed!";
                                cout << endl;
@@ -1543,16 +1544,18 @@

        void FFmpegVideoProvider::setSoundLevel(float level) {
                DFBResult retB;
+               DirectResult ret;

                if (level < 0.0) {
                        return;
                }

                if (rContainer->audio.playback) {
-                       retB = rContainer->audio.playback->SetVolume(
+                       // FIXME: DirectResult en vez de DFBResult
+                       ret = rContainer->audio.playback->SetVolume(
                                        rContainer->audio.playback, level);

-                       if (retB == DFB_OK) {
+                       if (ret == DR_OK) {
                                rContainer->volume = level;
                        }
                }
```

Una vez creado el archivo, se ejecuta el comando patch de la siguiente forma, es importante mencionar que el archivo se lo debe ejecutar desde la carpeta en la que se instaló el fusion sound, y consecuentemente el archivo de parche debe estar en la misma carpeta. En este caso, el directorio es dentro de 1.1.0:
Se ejecuta el siguiente comando:

```
patch -p0 < ginga.fusionsound.patch
```

Se presentarán dos lineas que verificaran el proceso del patch en la carpeta. Esas líneas deberían verse de esta manera:

```
patching file src/gingacc-system/src/io/interface/content/audio/FFmpegAudioProvider.cpp
patching file src/gingacc-system/src/io/interface/content/video/FFmpegVideoProvider.cpp
```

Compilación de Telemidia-Links

Acá debemos compilar e instalar un conjunto de librerías llamadas telemidia links. Como se explicó anteriormente es necesario leer bien lo que nos muestra al ejecutar los comandos, pues muchas veces

podrían dar error por falta de herramientas y librerías. En este caso en particular, nos debemos asegurar que esten instaladas las herramientas de OpenSSL y GPM. A continuación se detalla una lista de algunos de los paquetes más importantes que deben estar instalados. (En algunos casos, se requerirán de otros paquetes)

- ocaml-base-nox
- libfindlib-ocaml-dev
- libssl-dev
- camlp4
- libfindlib-ocaml
- libnewpki2

Una vez que nos aseguremos de los paquetes instalados, ejecutaremos el siguiente comando para el telemidia links:

```
sudo ./autogen.sh --enable-graphics --with-directfb --enable-javascript --without-x --without-sdl --prefix=/usr –enable-debuglevel=0
```

Cuando este termine de verificar, dará una lista resumen de las herramientas habilitadas. En el caso que no termine la verificación se debe leer con cuidado la herramienta que falta e instalarla. Posteriormente, se pasa a la compilación. Usamos los siguientes comandos:

```
sudo make
sudo make install
```

Después de este comando, se muestra un resumen de la instalación.
Compilación de Ginga

Ahora que hemos preparado todo, se procede a la instalación de Ginga propiamente dicho. Para esto nos vamos al directorio /usr/local. Una vez más cabe recalcar la necesidad de librerías adicionales para la ejecución del comando de compilación. Dado que este paso es el más importante se hará una lista con algunas librerías que se necesitan para la compilación:

- Sistema make, multiplataforma y de código abierto
- Boost C++ Libraries development files
- Gtest – Google's framework for writing C++ tests
- luabind C++ for LUA
- uniones libcURL para LUA 5.1
- libcurl-ocaml-dev
- cmake-dbg

Una vez dentro, ejecutamos los siguientes comandos.

```
cd 1.1.0/src/
sudo ./ginga-build.sh -i -S -P /usr/local -C /usr/local
```

Durante la instalación se mostrarán los paquetes instalados y los paquetes que faltan instalar. Se debe leer con cuidado para poder instalar las herramientas que faltan y completar la configuración y compilación del Ginga NCL.

Configuraciones Finales

Para terminar configurar los últimos detalles del software emulador de Ginga vamos a ejecutar algunos pasos finales. Creamos unos archivos de configuración. Primero debemos obtener permisos de administrador y ejecutar los siguientes comandos:

```
su
echo "/usr/lib/ginga" > /etc/ld.so.conf.d/ginga.conf
```

Luego cargamos nuestro archivo de configuración y nos salimos del modo administrador con los siguientes comandos:

```
ldconfig
exit
```

Luego creamos un archivo de configuración para el DirectFB, esto es necesario para poder trabajar sobre el entorno gráfico X11, Gnome, etc.

```
echo “system=x11” > ~/.directfbrc
```

Posterior a este comando deberíamos tener listo el Ginga.ar para ser ejecutado. Para poder probar un archivo en NCL simplemente ejecutamos la siguiente instrucción:

```
ginga --ncl <algún documento .ncl>
```