Instalación de Software GNU/Linux en Entornos Industriales

Manuel Espeleta García
ETIS

Miquel Àngel Senar

12/01/2011

Dedicatoria y agradecimientos

Dedicado a mi incondicional amiga Sandra, fiel compañera, querida hija y esperemos que pronto leal colega.

Agradecimientos a mi tutor Gerard Chiva por su esfuerzo en permitirme acceder a esta oportunidad a pesar de los inconvenientes.

Agradecimientos a mi consultor Miquel Àngel Senar por sus consejos.

Gracias.

Manuel Espeleta.

Resumen introductorio

Todo software creado para un desempeño industrial siempre tiene un comprometido factor común que es su instalación.

Entendiendo por software industrial el que se realiza para la industria con baja o nula distribución pública y muchas veces en entornos dedicados y hostiles.

Mientras que un software de distribución pública tiene un procedimiento de instalación mas o menos estándar, como pueden ser paquetes DEB o RPM y repositorios, en el caso de software para aplicaciones industriales no hay un sistema único.

Los elementos clave en este entorno suelen ser:

- Necesidad de una alta eficiencia en tiempo de instalación
- Minimizar los errores humanos
- Evitar el uso de Internet
- Máquinas nuevas y dedicadas

Este estudio evalúa las alternativas más comunes:

- Modificación de una distribución existente, *Ubuntu 10.04 LTS*
- Creación de una distribución propia desde fuentes, *LFS*
- Crear una distribución personalizada mediante aplicaciones *GNU/Linux*
- Uso de servicios web generadores de distribuciones a medida

Para la valoración de los métodos propuestos se ha dispuesto de un software creado para la ocasión y de máquinas virtuales en las que se han realizado las pruebas para verificar la operativa y su rendimiento.

Índice de contenido

Índice de ilustraciones

2 Memoria

2.1 Capítulo I - Introducción

El escenario usual del proceso de instalación de un programa en un entorno industrial suele tener varios denominadores comunes:

La máquina destino no dispone de sistema operativo, pues es nueva y dedicada para el software que se ha creado.

No se dispone de Internet, pues las comunicaciones de la máquina con el exterior están altamente restringidas o no dispone de comunicación.

El proceso de instalación debe ser muy veloz, pues aunque la generación del instalador de las pruebas pueden realizarse con cierta tranquilidad en un entorno de test, la instalación real se realiza en casa del cliente, requiere desplazamientos y autorizaciones.

El coste en los medios de instalación debe ser reducido, pues en algunos casos el cliente puede llegar a disponer de varios miles de máquinas y el entregar un soporte físico puede no ser viable.

Seguridad del proceso, pues una incidencia en el momento de la instalación puede provocar que todo el proyecto sufra una desviación y un descrédito delante del cliente.

Facilidad, pues es usual la dependencia con librerías específicas de terceros y procesos de propios de configuración y ajuste.

Cabe decir que el sistema ideal no existe dado que no todos los casos son iguales ni poseen las mismas restricciones ni presupuestos.

2.1.1 Justificación y contexto

En la actualidad las empresas que generan software industrial no tienen un sistema estándar de aplicar su creación a las máquinas destino, exceptuando la generación de paquetes DEB o RPM.

Este estudio evalúa los métodos usuales para la instalación del software bajo las condiciones usuales de los entornos industriales.

En mi experiencia profesional que se ha centrado en desarrollo de aplicaciones industriales en entornos Windows durante más de veinte años siempre he apreciado la carencia de soluciones específicas en los entornos GNU/Linux.

Si bien en la plataforma Windows existen aplicaciones profesionales de instalación, normalmente incluidas en los propios entornos de generación de aplicaciones, no se aborda la cuestión de generar una instalación de una máquina nueva. Adicionalmente la complejidad de estas aplicaciones hace que su uso sea limitado.

Este trabajo aporta información sobre las ventajas he inconvenientes de los diferentes sistemas de y ha de ser de ayuda a quienes quieran minimizar los costes de instalación. También se aporta una solución mediante scripts que auna los objetivos demandados.

2.1.2 Objetivos

El objetivo es aportar un sistema de instalación en entornos industriales que de solución a los retos que se plantean en esos escenarios. A través del estudio y pruebas de las opciones más comunes y actuales se determina las ventajas e inconvenientes de cada procedimiento.

También es objeto de este trabajo dar una visión de los diferentes procedimientos para llevar a cabo la generación de un instalador de estas características.

Este documento puede considerarse como el fin mismo del proyecto, pues aporta las necesidades que han de cumplir los procedimientos, la explicación de cada uno, las pruebas realizadas una comparativa, las conclusiones a las que se ha llegado, siempre sujetas a las premisas planteadas, y lo que es más útil, el script de instalación que resuelve el problema planteado.

Además de recomendar un sistema basado en las pruebas, los requisitos y resultados también es una finalidad de esta memoria dar una visión de las alternativas, pues pueden ser de mucha utilidad para escenarios con otras características.

2.1.3 Enfoque y método del trabajo

Desde una óptica empresarial orientada a resultados se han revisado los sistemas bajo estudio. Es común que la parte del presupuesto destinada a la instalación siempre es la mínima imprescindible, pues no ofrece un beneficio directo. Por tanto los costes del sistema, en aprendizaje, en implementación, son tan importantes como el resultado en si mismo.

El trabajo se ha estructurado en una primera fase de recopilación de información, esencialmente mediante recursos de internet. En esta fase se han recabado las alternativas y el funcionamiento de cada procedimiento.

Posteriormente se ha generado un programa para las pruebas. Simulando las necesidades usuales es una programa con dependencia de librerías que no se incorporan en las distribuciones usuales. En particular dicho programa tiene dependencias de librerías OpenGL pero se ha de entender como una dependencia arbitraria no satisfecha por una distribución estándar.

Se han creado diversas máquinas virtuales mediante VirtualBox 3.2 sobre las que se han efectuado las pruebas, tanto como generadoras de la solución como 'maquinas nuevas' en las que se ha probado el resultado. De esta forma se ha probado tanto el procedimiento como los resultados con las mínimas interferencias provocadas por el uso de un sistema modificado.

La distribución de partida es Ubuntu 10.04 LTS por su gran distribución, probada estabilidad y con soporte de larga duración. Sus características la hacen recomendable para cualquier sistema industrial.

Como punto de referencia para la comparación de tiempos la máquina usada ha sido un Intel i7 950, 12 Gb RAM, SSD 60 Gb y HD 2Tb y las máquinas virtuales se han creado definiendo 3 procesadores, 1 Gb RAM y HD 12 Gb, este último siempre creado sobre el HD 2 Tb.

2.1.4 Planificación del proyecto

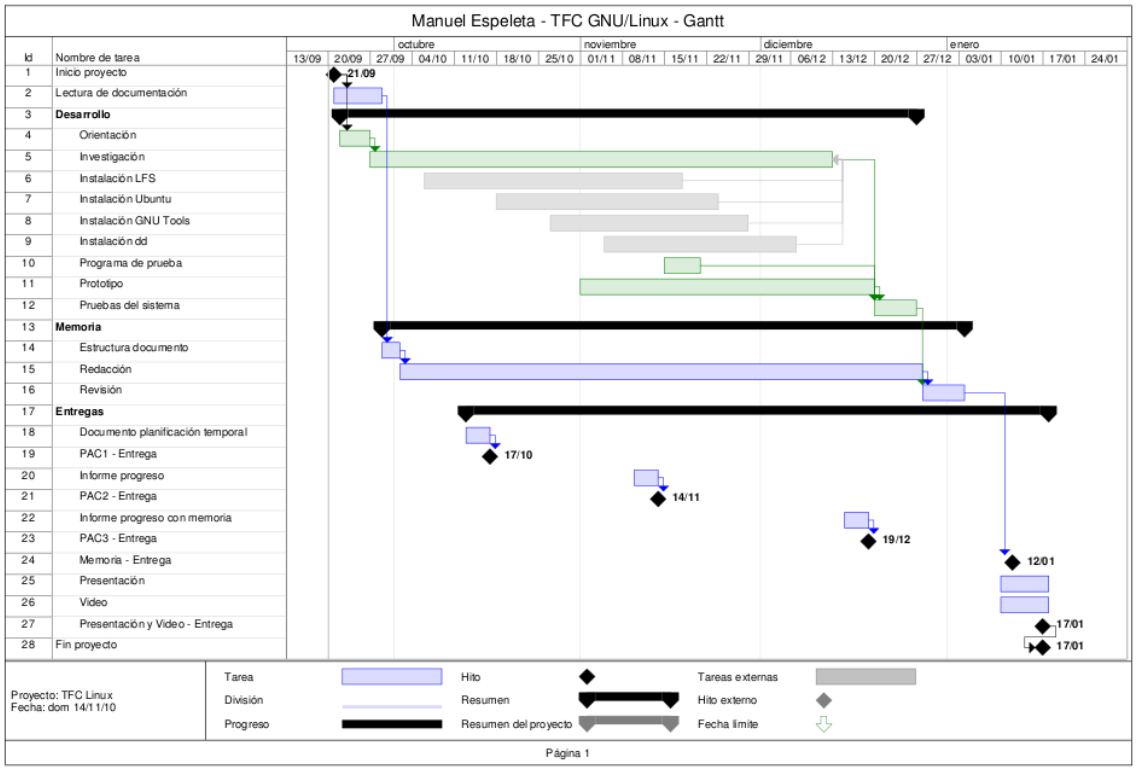

Ilustración 1: Gantt de planificación del proyecto

El proyecto se ha planificado como una tarea de investigación y pruebas, a la par que se generan los documentos requeridos.

Hay una gran fase en la que se recaba información de los sistemas de instalación que es el cuerpo del proyecto. Al tiempo se realizan las tareas de documentación.

Durante la fase de desarrollo también se ha previsto la realización del programa de prueba, el prototipo.

Los procesos de prueba y verificación son implícitos a la finalización de cada fase.

Las tareas de entrega parciales se han realizado de forma paralela a las propias del proyecto, tal y como se aprecia en el Gantt adjunto.

2.1.5 Productos obtenidos

Como producto objetivo en el Anexo II se encuentra en el anexo los scripts necesarios para generar una instalación software. Estos scripts generan un LiveCD totalmente autónomo, que contiene el programa que se desee instalar y las librerías necesarias.

El producto ofrecido es totalmente funcional y permite realizar la configuración necesaria. El procedimiento es fácilmente ampliable a casos que requieran de más detalle, pues el proceso que realizan los scripts es fácil de modificar y permite con unos mínimos cambios operar manualmente en todos los ajustes propios del sistema operativo.

También, y como producto adicional, se ofrece una documentada valoración de los diferentes sistemas con sus ventajas e inconvenientes, con una opción destacada sobre el resto. Esta información esta disponible en el capítulo Conclusiones.

2.1.6 Otros capítulos

En el siguiente capítulo está el estudio de cada sistema de instalación. Cada estudio engloba el trabajo realiza sobre cada sistema, desde la instalación a las impresiones iniciales y puede considerarse completo e independiente para una revisión de un procedimiento de instalación.

Le sigue el capítulo de una valoración de costes, una comparativa de los diferentes procedimientos. Se evalúa cada sistema desde un punto de vista de recursos y beneficios, pues las herramientas usadas no requieren un desembolso económico o no es destacable.

El capítulo de conclusiones ofrece una opinión enfocada desde diversos puntos de vista. Puede enfocarse como una guía rápida para quien quiera ver el resultado rápidamente.

2.2 Capítulo II - Métodos de instalación

En este capítulo se abordan los métodos y su funcionamiento.

Para cada método se muestra una introducción, los prerrequisitos, el procedimiento, el resultado y una valoración.

- En la introducción se explica el punto de partida y una visión global.
- Los prerrequisitos citan los elementos necesarios para llevar a cabo el proceso.
- El procedimiento detalla el proceso para llevar a cabo la instalación.
- El resultado informa del producto resultante.
- En la evaluación se destacan los puntos fuertes y débiles del sistema, dejando para el capítulo 'Conclusiones' el contraste con los demás sistemas.

2.2.1 Modificación de una distribución, *Ubuntu 10.04 LTS*

2.2.1.1 Introducción

Ubuntu dispone de instrucciones propias para la personalización de su LiveCD. Ver https://help.ubuntu.com/community/LiveCDCustomization

Siguiendo los pasos indicados se consigue un CD de arranque que instala el sistema operativo con los programas que se deseen añadir.

El proceso se basa en extraer el contenido del LiveCD en un directorio, modificarlo y generar una nueva imagen.

2.2.1.2 Prerrequisitos

Para el proceso se requiere una máquina con Ubuntu 6.06 o posterior, estas poseen el soporte necesario de *squasfhs*, y las herramientas instaladas *squashfs-tools* y *genisoimage*.

Si se dispone de la máquina con Ubuntu e Internet, las herramientas citadas se pueden instalar con facilidad desde el propio gestor de paquetes de la distribución o bien mediante el comando:

```
sudo apt-get install squashfs-tools genisoimage
```

También se ha de disponer de una imagen del LiveCD de insalación y de espacio suficiente para el proceso, se recomienda un espacio libre mínimo de 3 a 5 Gb.

En las pruebas realizadas se ha partido de:

- Máquina virtual con sistema operativo Ubuntu 10.04 LTS 32 bits
- Imagen LiveCD Ubuntu 10.04 LTS 32 bits
- Paquete squashfs-tools instalado, versión 1:4.0-6ubuntu1
- Paquete genisoimage instalado, versión 9:1.1.10-1ubuntu1
- 4 Gb de espacio libre

2.2.1.3 Procedimiento

El proceso parte de la imagen LiveCD. Esta se copia a un directorio de trabajo y se extrae su contenido para poder realizar los cambios. En el proceso concreto de pruebas se han añadido los paquetes adicionales en el mismo directorio de trabajo.

Una vez que se dispone de la imagen expandida es posible realizar los cambios oportunos para la generación de la nueva imagen, en este punto es especialmente interesante la inclusión, o eliminación, de paquetes entre los que se pueden incluir los propios y sin necesidad de repositorios.

Finalmente hay que realizar unos sencillos procesos de reconstrucción y se genera la imagen.

Seguidamente se detalla el proceso y los comandos usados. En el Anexo II se adjuntan los scripts para la automatización del proceso, con ellos sólo se requiere teclear el inicio y las líneas indicadas por el propio script, tal y como se aprecia en la ilustración adjunta. Mediante estos scripts se evitan errores al teclear y se mejora el tiempo de ejecución.

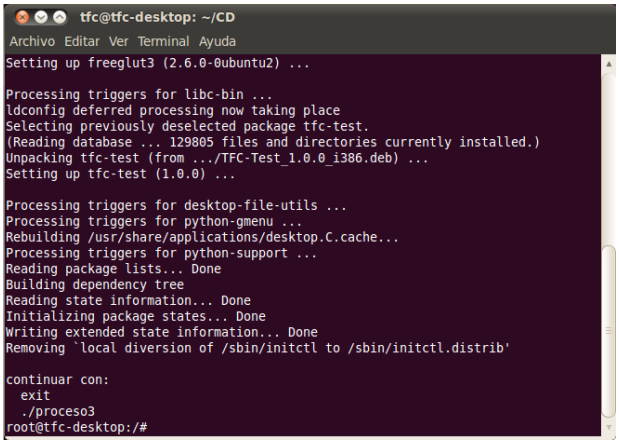

Ilustración 2: Scripts de instalación

Copiado de la imagen al directorio de trabajo

```
mkdir -p ~/CD
cp ubuntu-10.04-desktop-i386.iso ~/CD
cd ~/CD
```

Montado y extracción del contenido de la imagen

```
mkdir -p mnt
sudo mount -o loop ubuntu-10.04-desktop-i386.iso mnt
mkdir -p extract-cd
rsync --exclude=/casper/filesystem.squashfs -a mnt/ extract-cd
```

Extracción del entorno de escritorio

```
sudo unsquashfs mnt/casper/filesystem.squashfs
sudo mv squashfs-root edit
```

Copia de los paquetes propios a instalar (variar según ubicación origen)

```
sudo cp *deb edit/var/cache/apt/archives
```

Cambio de entorno

```
sudo mount --bind /dev/ edit/dev
sudo chroot edit
```

Preparación del nuevo entorno (se visualizará un nuevo indicador del sistema)

```
mount -t proc none /proc
mount -t sysfs none /sys
mount -t devpts none /dev/pts
export HOME=/root
export LC_ALL=C
```

Llegado este punto ya se dispone de un entorno modificable en el que se realizaran las personalizaciones. La cantidad de modificaciones realizable es tan amplia como se desee, desde el fondo de la pantalla a la creación de usuarios. En el caso particular que se muestra se realizará la instalación de los programas adicionales.

Pasos iniciales

```
dbus-uuidgen > /var/lib/dbus/machine-id
dpkg-divert --local --rename --add /sbin/initctl
ln -s /bin/true /sbin/initctl
```

Instalación de programas adicionales copiados anteriormente

```
dpkg -i /var/cache/apt/archives/freeglut3_2.6.0-0ubuntu2_i386.deb
dpkg -i /var/cache/apt/archives/TFC-Test_1.0.0_i386.deb
```

Limpieza del nuevo entorno.

```
aptitude clean
rm -rf /tmp/* ~/.bash_history
rm /var/lib/dbus/machine-id
rm /sbin/initctl
dpkg-divert --rename --remove /sbin/initctl
```

Salida del nuevo entorno.

```
umount /proc
umount /sys
umount /dev/pts
exit
sudo umount edit/dev
```

Regenerar la lista de paquetes

```
chmod +w extract-cd/casper/filesystem.manifest
sudo chroot edit dpkg-query -W --showformat='${Package} $
{Version}\n' > extract-cd/casper/filesystem.manifest
sudo cp extract-cd/casper/filesystem.manifest extract-
```

```
cd/casper/filesystem.manifest-desktop
sudo sed -i '/ubiquity/d' extract-cd/casper/filesystem.manifest-
desktop
sudo sed -i '/casper/d' extract-cd/casper/filesystem.manifest-
desktop
```

Crear el nuevo sistema de ficheros

```
sudo rm -f extract-cd/casper/filesystem.squashfs
sudo mksquashfs edit extract-cd/casper/filesystem.squashfs
```

Personalizar el fichero de identificación del CD (opcional)

```
sudo nano extract-cd/README.diskdefines
```

Actualizar sumas MD5

```
cd extract-cd
sudo rm md5sum.txt
find -type f -print0 | sudo xargs -0 md5sum | grep -v
isolinux/boot.cat | sudo tee md5sum.txt
```

Generación de la nueva imagen

```
sudo mkisofs -D -r -V "UOC" -cache-inodes -J -l -b
isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -boot-load-
size 4 -boot-info-table -o custom.iso .
```

2.2.1.4 Resultado

Al finalizar se obtiene una imagen de CD que se puede grabar directamente sobre un soporte, o en las pruebas utilizar directamente como imagen de inicio de las máquinas virtuales.

Como se aprecia en la ilustración esta imagen de CD proporciona el mismo proceso de instalación del CD original, el propio de Ubuntu. De esta forma la configuración y ajustes propios de la máquina destino los realiza el software de Ubuntu.

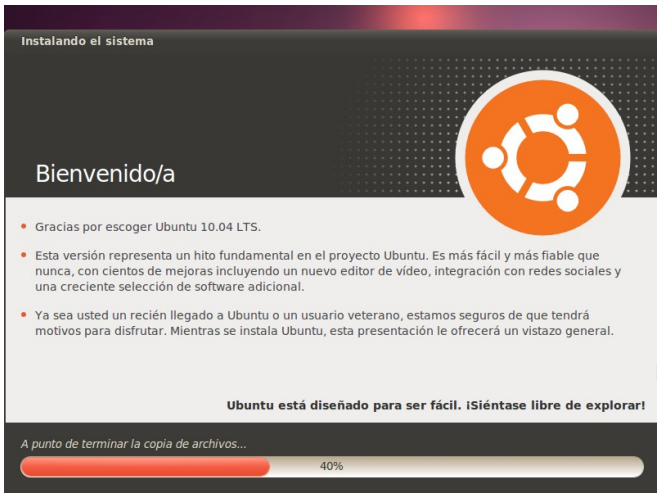

Ilustración 3: Proceso de instalación LiveCD personalizado

Al finalizar el proceso de instalación mediante el CD generado aparece el escritorio propio de la instalación original pero con la aplicación ya instalada. En el caso concreto del prototipo usado la nueva máquina arranca con la aplicación en primer plano sin realizar ninguna acción adicional.

Cabe destacar que el resultado incluye el software prototipo, la configuración del mismo, su inicio automático al iniciarse el entorno de ventanas y las librerías necesarias ya instaladas.

La ilustración adjunta muestra la pantalla del equipo tras el primer inicio al finalizar la instalación, sin intervenciones adiciones.

Ilustración 4: Arranque del sistema instalado con LiveCD personalizado

2.2.1.5 Evaluación

Es un procedimiento que destaca por su velocidad, sencillez y resultado.

- Es un sistema estable desde hace 4 años y soporte de la comunidad de Ubuntu.
- Es un proceso relativamente fácil y rápido.
- El sistema resultante es tan compatible como la distribución.
- El trabajo de configuración posterior se puede reducir a cero.

La creación del CD es un proceso rápido, su duración es muy dependiente de la máquina usada pero se mide en minutos.

Ha modo de referencia el proceso de extracción de la imagen ha tenido un tiempo inferior a los dos minutos, mientras que el proceso de generación de la ISO ha sido de tres minutos.

Una vez generado el CD el tiempo de instalación sobre una máquina nueva es el usual del sistema operativo, destacablemente rápido en Ubuntu sobre un disco de reducidas dimensiones. Pero es de resaltar que al finalizar la instalación también termina el tiempo invertido pues el sistema ya es operativo.

Una vez probado el sistema se aprecian las posibilidades adicionales como la eliminación de paquetes, cambios del interface (como el fondo del escritorio) o creación de usuarios. Todas ellas pueden quedar ajustadas al finalizar la instalación del CD, aunque cabe decir que si bien no son complejos estos ajustes adicionales tampoco con triviales.

Ventajas

- Velocidad de preparación
- Velocidad de instalación
- Eliminación de ajustes manuales post instalación
- Alta compatibilidad con elementos Hardware

Desventajas

- Dependencia relativa de la instalación origen

2.2.2 Creación de una distribución desde cero, *LFS*

2.2.2.1 Introducción

La idea inicial fue generar una distribución desde los fuentes, *LFS* o *Linux From Scratch* como se suele llamar.

Esta solución tiene unos inconvenientes que se hicieron evidentes nada más empezar el estudio. Requiere de unos conocimientos muy elevados de toda la estructura, pues una distribución por sencilla que sea incorpora un gran número de elementos con configuraciones específicas para cada uno de sus componentes. También tiene el inconveniente que se han de generar unos scripts de gran tamaño, y con una gran dependencia con las versiones de todos los módulos empleados lo que implica un coste de mantenimiento muy alto.

Sin embargo el mayor escollo para implementar una distribución de este tipo es la dificultad de seguir los estándares.

Sin contar el núcleo hay dos elementos clave en toda distribución, el arranque y la gestión de procesos. La mayor parte de la información obtenida para la creación de una distribución desde cero esta obsoleta a día de hoy.

Los cambios más destacados en las distribuciones han sido el arranque que ha pasado de LILO a Grub y de Grub a Grub2 muy recientemente y por otra parte el proceso de gestión de procesos a SystemV a Upstart también recientemente.

El entorno gráfico ha sufrido variaciones con la mejora de las versiones de los entornos, como Gnome o KDE, y se anuncian más variaciones para el mejor uso de las GPU de las máquinas actuales.

Dado que LFS parte de los fuentes originales estos cambios implican no sólo un esfuerzo en la adaptación si no una dificultad en la integración de todos los elementos.

Se ha usado la última distribución en el momento del estudio que es la 6.7

2.2.2.2 Prerrequisitos

- Sistema anfitrión con los siguientes paquetes:
  - ✔ Bash-3.2
  - ✔ Binutils-2.17
  - ✔ Bison-2.3
  - ✔ Bzip2-1.0.4
  - ✔ Coreutils-6.9
  - ✔ Diffutils-2.8.1
  - ✔ Findutils-4.2.31
  - ✔ Gawk-3.1.5
  - ✔ Gcc-4.1.2
  - ✔ Glibc-2.5.1
  - ✔ Grep-2.5.1a
  - ✔ Gzip-1.3.12
  - ✔ Linux Kernel-2.6.22.5 (compilado conGCC-4.1.2 o posterior)
  - ✔ M4-1.4.10
  - ✔ Make-3.81
  - ✔ Patch-2.5.4
  - ✔ Perl-5.8.8
  - ✔ Sed-4.1.5
  - ✔ Tar-1.18
  - ✔ Texinfo-4.9

  Nota: es posible realizar el LFS con otras versiones. Pero no hay garantías sobre su funcionamiento y pueden requerir cambios en el proceso. Especialmente si son anteriores a las citadas.

- Disponer de una partición o disco de más de 1,3 Gb disponibles

- Libro electrónico Linux From Scratch 6.7

  Este documento detalla todo el proceso de instalación incluyendo los comandos necesarios.

  La instalación no esta basada en scripts por lo que se han de introducir los comandos según se realiza. Hay alternativas basadas en LFS con scripts pero no están tan actualizadas. Por lo que hay que seguir el libro indicado, 322 páginas, mientras se realiza el proceso.

- Paquetes fuente y actualizaciones del software a instalar

  Son 59 paquetes fuente y 19 actualizaciones en esta versión. Todos ellos han de ser la versión exacta indicada en el libro LFS 6.7. El volumen de descarga son 281 Mb en formato comprimido, 1 Gb expandidos.

Tanto el libro como el software necesario puede ser descargado desde http://www.linuxfromscratch.org/mirrors.html

2.2.2.3 Procedimiento

Para realizar la instalación inicial se requiere de un sistema operativo anfitrión desde el que se realiza la compilación de los componentes.

Se recomienda que el sistema operativo anfitrión no sea una máquina en explotación por las posibilidades de dañarlo.

El entorno recomendado es que se aprecia en la imagen adjunta, una máquina virtual desde la que se realiza el proceso con el libro y la consola a la par.

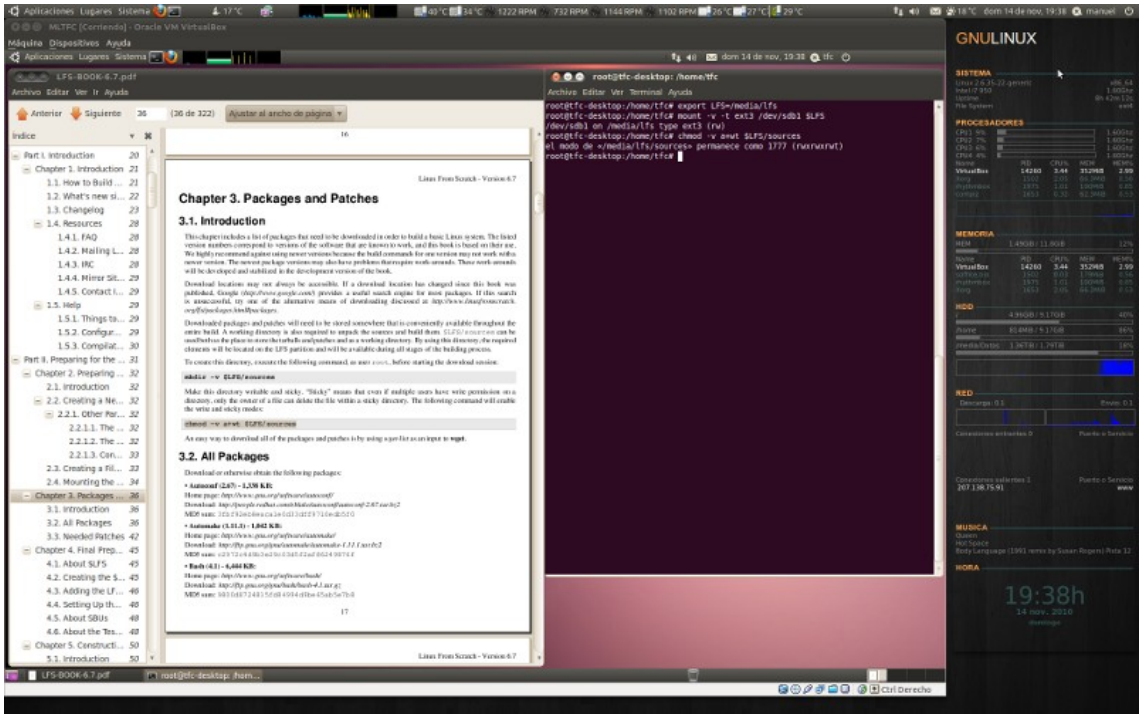

Ilustración 5: Entorno de trabajo con LFS

Es posible usar como entorno de trabajo el propio LiveCD pero se desaconseja, pues no esta previsto para él y requiere de cambios en el proceso, provocados por la estructura y la imposibilidad de grabar. Tampoco se disponen de herramientas que faciliten el trabajo como copiar y pegar desde el libro (aunque se pueden instalar, desaparecen en cada reinicio de la máquina).

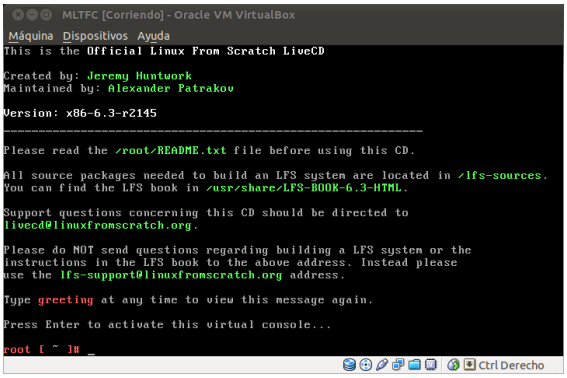

Ilustración 6: LFS mediante LiveCD

La compilación de la imagen oscila desde unas horas a unos días según la máquina usada. Sin embargo dado que se requiere teclear según se avanza el tiempo requerido es claramente superior al de compilación.

La configuración del sistema una vez compilado se ajusta modificando manualmente los scripts de inicio. En este apartado hay que hacer variaciones sobre el libro para la internacionalización del sistema.

2.2.2.4 Resultado

El proceso citado deja un sistema operativo perfectamente adaptado a la máquina sobre el que personalizar la instalación del software.

Cabe destacar que el resultado no tiene soporte gráfico más allá de Ncurses. Por lo que si se requiere un entorno gráfico se requiere de trabajo adicional.

La instalación de software propio sobre el entorno generado es trivial en comparación del proceso realizado, siempre y cuando no se necesiten más librerías.

2.2.2.5 Evaluación

Lo más destacado:

- *LFS* no esta al alcance de todo el mundo por los conocimientos requeridos.
- Es un proceso largo y laborioso con grandes probabilidades de error.
- No da soporte para elementos básicos como el entorno gráfico.
- No hay un sistema de actualizaciones o de incorporación de nuevo software mas allá de realizarlo manualmente.

Pensar que se puede generar una distribución base monolítica y que permanezca inmutable para nuestros propósitos es totalmente desaconsejable, pues los proveedores de librerías o hardware necesario en el entorno industrial pueden llegar a estar obligados a cumplir los nuevos estándares pero no tienen porque dar soporte a sistemas obsoletos.

Los cambios en las distribuciones recientes son suficientemente elevados para desestimar esta solución. El riesgo de error en el proceso también la hacen poco viable, a lo que hay que sumar la escasa portabilidad o el tiempo necesario.

En máquinas destinadas a servidores de alto rendimiento puede ser una opción aunque hay instalaciones como Gentoo que facilitan esta tarea con el mismo resultado a nivel de optimización.

Ventajas

- Minimización del producto objetivo
- Alta optimización

Desventajas

- Complejidad excesiva
- Coste de mantenimiento elevado
- Poco portable por compilación orientada a la optimización
- Desviación de los estándares por los profundos cambios en las distribuciones
- Sin soporte gráfico

2.2.3 Distribución personalizada con herramientas GNU/Linux

2.2.3.1 Introducción

Una primera aproximación para crear un LiveCD propio es usar alguna de las herramientas GNU/Linux diseñadas para este fin.

Existen varias alternativas pero con enfoques dispares:

Remastersys

Proyecto GNU/Linux multipropósito que permite crear copias de seguridad e del sistema a CD.

Genera un CD copia del sistema aunque tiene algunas limitaciones. La imagen ISO generada es considerablemente mayor y no permite la inclusión de paquetes adicionales. De todas formas en las pruebas ha creado un LiveCD con instalador funcional.

Si bien es un proyecto mantenido al día se denota el carácter particular pues los repositorios no están actualizados, el manual es cedido por otra web y no está actualizado.

http://www.geekconnection.org/remastersys

Fedora Pungi y livecd-creator

Ofrece la generación de un CD basado en la elección de RPM de Fedora.

La orientación de esta utilidad no coincide con el objetivo de crear un instalador industrial, pues no se destina a añadir paquetes propios sino a escogerlos de entre los que Fedora propone, aunque propone la modificación de los scripts que integran el proceso.

Cabe destacar que se han encontrado muchos enlaces rotos dentro del propio web de Fedora que denotan un inmaduro estado del proyecto.

http://fedoraproject.org/wiki/How_to_create_and_use_a_Live_CD

Reconstructor

Reconstructor es una herramienta de pago basada en un entorno Javascript para la generación de LiveCD personalizados.

Es una herramienta estable que permite la inclusión de paquetes DEB y de repositorios adicionales.

https://reconstructor.apphosted.com

UCK - Ubuntu Customization Kit

Script de personalización del LiveCD de Ubuntu.

Es una alternativa similar a la presentada en este documento mediante un entorno gráfico pero requiere de modificación de script manualmente para incluir paquetes adicionales. Este paso se basa en repositorios externos sin indicaciones para instalación local aunque es posible añadirlos.

Es un reconocido proyecto en activo, aunque su enfoque dista del que se evalúa aquí centrándose en cambios del operativo. El proceso de prueba se detuvo sin generar la imagen de CD.

http://uck.sourceforge.net/

Garfio

Muy similar al anterior, UCK, aunque más genérico.

Tampoco esta enfocado a la incorporación de nuevo software.

Posee una web propia y el proyecto está mantenido al día.

http://www.garfio.org.ar/

Otras herramientas (Scripts)

http://www.linux-live.org/

http://www.gnewsense.org/Builder?action=show&redirect=Builder%2FBuilder

Cada herramienta dispone de un enfoque distinto aunque todas son capaces de generar LiveCD mas o menos personalizado. En este se caso se ha elegido Remastersys por su carácter gráfico, amigable y funcional.

2.2.3.2 Prerrequisitos

Para usar esta aplicación se requiere añadir el repositorio:

```
deb http://www.geekconnection.org/remastersys/repository karmic
```

Y se instala el paquete remastersys desde el gestor de paquetes. Para las distribuciones más modernas como jaunty o maverick se sigue usando el repositorio indicado. Tampoco se dispone de información para la autenticación.

Es necesario disponer de espacio libre para la generación. Se recomienda un mínimo de 5 Gb.

2.2.3.3 Procedimiento

La aplicación se inicia en el menú Sistema → Administración → Remastersys Backup.

Tal y como aparece en la ilustración adjunta sólo es necesario indicar la opción del programa. Con 'Dist' generará una imagen ISO con instalador copia del entorno en que se ejecuta.

El proceso dura unos seis minutos en la máquina de pruebas.

Al acabar se indica el lugar en el que se encuentra la imagen, lista para copiar a CD.

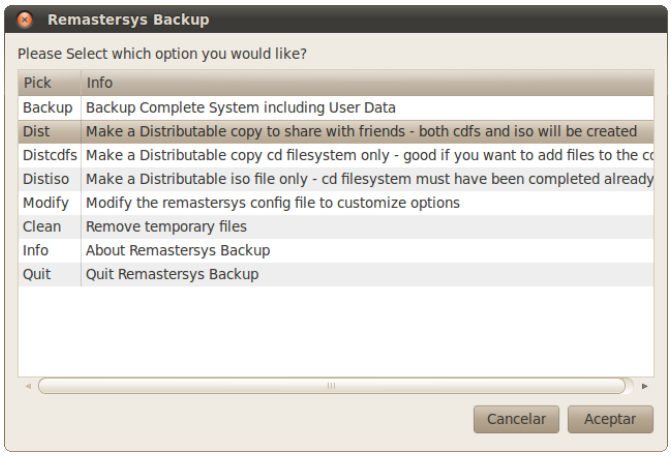

Ilustración 7: Selección de opciones en Remastersys

2.2.3.4 Resultado

Es un sistema simple, rápido y eficaz.

En las pruebas realizadas sobre un entorno previamente configurado se generó el instalador y con él se generó la instalación destino sin contratiempos. La máquina resultante era una aparente copia de la original en cuanto a funcionalidad, pero con la instalación propia del sistema operativo.

2.2.3.5 Evaluación

Lo más destacado:

- Simple y funcional.
- Posee limitaciones de configuración.
- Dependencia de un desarrollo no profesional.
- Posible incompatibilidad con sistemas futuros.

Sin duda es una herramienta a tener en cuenta. Más allá de una simple copia el hecho de incorporar el instalador ofrece una compatibilidad muy amplia.

Destaca que no es necesaria ninguna imagen de CD, lo que parece indicar una cierta dependencia con la instalación.

Es una buena opción si se dispone de una máquina de pruebas limpia en la que se ha instalado el software que se quiere distribuir.

Ventajas

- Fácil y rápida
- Instalación compatible.

Desventajas

- Proyecto no profesional
- Posible dependencia con versiones concretas
- Tamaños de imagen más grandes de lo habitual

2.2.4 Servicios web generadores de distribuciones a medida

2.2.4.1 Introducción

Suse causó impacto en el mundo GNU/Linux al ofrecer descargar una ISO a medida. En unos pocos minutos de ajustes en su página web y de enviar los ficheros a incluir esta web es capaz de generar un instalador a medida.

http://susestudio.com/

2.2.4.2 Prerrequisitos

Los ficheros a instalar han de servirse como paquetes de instalación RPM.

2.2.4.3 Procedimiento

Una vez registrado en la web el proceso consiste en la selección de las opciones que definen la imagen. En la ilustración adjunta se aprecian los menús que definen los pasos a realizar.

El tiempo invertido osciló de 6 a 15 minutos para la construcción de la imagen y de 14 a 40 para la descarga. En las pruebas se usó una conexión ADSL estándar de 10 Mbits efectivos.

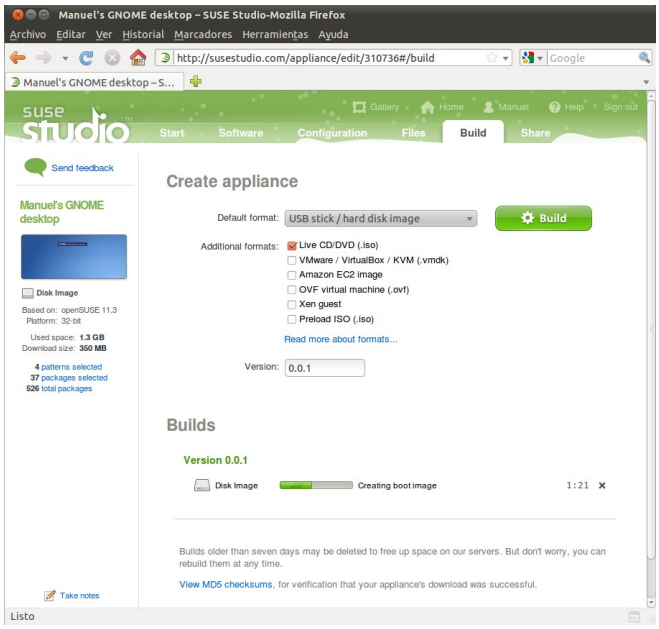

Ilustración 8: Personalización de SUSE

El procedimiento es:

- Dar un nombre a la imagen.
- Seleccionar los paquetes que conforman la distribución.
- Enviar los RPM adicionales a instalar.
- Configurar el entorno, incluyendo idioma, logo, fondo, Red, usuario, inicio gráfico, base de datos, acceso log in automático, inicio de programas, ajustes de disco y de la máquina virtual y sripts adicionales.

- Enviar ficheros adicionales, si procede.
- Seleccionar el tipo de generación, ISO en este caso.
- Crear la distribución y descargarla. Antes de descargar la distribución es posible probarla en el propio servidor de Suse.

2.2.4.4 Resultado

Es un sistema eficaz y completo. Aunque genera algunos recelos al plantearse como sistema empresarial.

En primer lugar la licencia necesaria se debe estudiar con detenimiento, en el caso de las pruebas es una licencia no restrictiva, pero en el caso de ser una empresa se debe considerar.

Con la suscripción se permiten 15 Gb de datos, más que suficiente. Quizás preocupe más depender de una conexión y de la funcionalidad de un servicio remoto.

2.2.4.5 Evaluación

Lo más destacado:

- Simple y funcional
- Configuración detallada del sistema
- Dependencia de un servicio externo remoto

Sin duda es una herramienta a tener en cuenta. Más allá de una simple copia el hecho de incorporar el instalador ofrece una compatibilidad muy amplia.

Destaca que no es necesaria ninguna imagen de CD, lo que parece indicar una cierta dependencia con la instalación.

Es una buena opción si se dispone de una máquina de pruebas limpia en la que se ha instalado el software que se quiere distribuir.

Ventajas

- Simple y funcional
- Instalación compatible
- Alto nivel de personalización
- Reducido tamaño de la imagen

Desventajas

- Dependencia de un servicio externo y remoto
- Dependencia estricta con SUSE

2.3 Capítulo III - Valoración de costes

Como se ha comentado y por la particularidad del proyecto los costes se basan en los esfuerzos derivados de su uso. El coste se expresa según los siguientes factores:

- Complejidad. La complejidad eleva los recursos necesarios.
- Tiempo de generación de la instalación. El tiempo consumido en la generación no es un factor determinante pero si es a tener en cuenta, pues puede ser necesario ejecutarlo múltiples veces.
- Tiempo de instalación. Es el objetivo del instalador, que realice su tarea en el mínimo tiempo.
- Compatibilidad del proceso. La compatibilidad garantiza menos incidencias en su uso y por tanto menos costes.
- Compatibilidad del producto. Similar a la anterior pero en el momento de la instalación.
- Dependencias externas. Si el procedimiento tiene mucha dependencia de terceros puede implicar que se tenga que adaptar a cambios o limitaciones.
- Fiabilidad de la instalación. Todas las garantías son pocas en un proceso tan trascendental de un proyecto.

	Ubuntu	**LFS**	**GNU Tools**	**Web**
Complejidad	Baja	Alta	Mínima	Mínima
Tiempo generación	Muy bajo	Muy Alto	Muy bajo	Bajo
Tiempo instalación	Bajo	Bajo	Bajo	Bajo
Compatibilidad proceso	Alta	Baja	Media	Media
Compatibilidad producto	Alta	Baja	Alta	Alta
Dependencias externas	Baja	Alta	Media	Media
Fiabilidad	Alta	Baja	Alta	Alta

Se aprecia claramente que el procedimiento LFS tiene un coste desmedido, mientras que los demás sistemas son perfectamente asumibles.

2.4 Capítulo IV - Conclusiones

A pesar de la disparidad de opciones hay buenas alternativas.

Las pruebas han mostrado diversos procedimientos viables. A destacar el buen resultado conseguido, en muchos casos en forma de LiveCD con instalador externo con amplio soporte de hardware.

La elección de uno u otro método puede recaer en las preferencias o necesidades puntuales. Para más detalles ver el capítulo 'Valoración de costes' o el estudio del procedimiento elegido.

Modificación de una distribución, Ubuntu 10.04 LTS

Esta es la opción recomendada, si bien es cierto que por poco margen y según preferencias.

Dejando a LFS a parte, es una opción ligeramente más compleja pero permite un mayor control y una menor dependencia que el resto. Cumple perfectamente los requisitos y solventa la dificultad de la generación sin uso de repositorios externos.

Aunque el sistema no implica una total dependencia con Ubuntu, sino sólo con las herramientas estándar de generación, también es cierto que es el entorno al que se limita inicialmente. De todas formas hoy por hoy la elección de Ubuntu es recomendable cuando menos.

Creación de una distribución desde cero, LFS

Totalmente desaconsejable en entornos empresariales en los que priman la efectividad, la compatibilidad y el resultado.

Puede ser un buen ejercicio para estudio o para sistemas muy especiales. A destacar que permite la generación de instaladores para máquinas distintas a las de generación del instalador, lo que permite usarlo en sistemas no estándar.

Como ventajas también destaca lo minimalista del producto y su optimización, aunque no son determinantes en las premisas del entorno empresarial.

Distribución personalizada con herramientas GNU/Linux

Existe un amplio abanico de opciones que cubren desde los scripts a los programas automatizados. Remastersys, la opción elegida, ha dado unos resultado destacables. En general en estas herramientas se hecha en falta un buen control sobre los paquetes a instalar de forma local durante la generación.

Remastersys es un sistema de generación muy cómodo y con muy buen resultado. La dependencia con el proyecto, que muestra carencias de mantenimiento, es quizás su mayor inconveniente.

Servicios web generadores de distribuciones a medida

La propuesta se centra en SUSE, pues no se han encontrado alternativas que puedan competir con él en el apartado web.

Es una solución innovadora, eficaz y cómoda, que al igual que Remastersys peca de su dependencia externa. En el caso de SUSE también se limita el operativo destino, en este caso a uno de menor difusión que Ubuntu.

Sin duda es una buena elección para realizar unas rápidas pruebas del procedimiento de instalación, pues permite tanto instalar paquete como realizar ajustes pormenorizados con total facilidad. Sin embargo es preferible que un departamento de software tenga sus propias herramientas de forma local si es posible.

Notas finales

La personalización de Ubuntu aquí mostrada no es sino una alternativa GNU/Linux más, si bien cubre mejor las premisas pues se ha creado con ese objetivo.

Las alternativas, LFS a parte, suelen tropezar con la inclusión de software local o con dependencias con terceros, siempre a valorar por cada interesado y según su entorno.

3 Glosario

DEB

Sistema de agrupación de software que permite su fácil instalación en sistemas derivados de la distribución Debian.

Debian

Una de las más conocidas distribuciones GNU/Linux, reconocida por su estabilidad de funcionamiento. Creada y mantenida por una comunidad de programadores nació como un impulso del software libre.

Fedora

Nacio como vertiente abierta de la distribución Red Hat de la empresa Red Hat Enterprise Linux.

Gentoo

Conocida distribución basada en la compilación de los paquetes.

GNU/Linux

Referencia a la combinación del núcleo de Linux con las herramientas GNU, que básicamente son aplicaciones con licencia GPL para su libre uso, modificación y distribución.

LFS

Acrónimo de Linux From Scratch. Es un proyecto GNU/Linux para la difusión del proceso de creación de una instalación del operativo Linux completo desde los ficheros fuente de sus componentes. El proceso requiere la compilación de todos los módulos integrantes y ajustes manuales.

Live CD

Disco óptico que contiene una distribución Linux funcional iniciable desde la unidad óptica. Este sistema permite probar e incluso trabajar con la distribución sin afectar a la máquina que lo usa, aunque pueden realizarse cambios si se habilitan sus discos de almacenamiento desde la distribución manualmente.

LTS

Acrónimo de Long Time Support que usas Ubuntu para distinguir las versiones que se emiten cada dos años. Éstas tienen un soporte de más de dos años.

Red Hat

Ver Fedora.

Remastersys

Desarrollo personal de Tony Brijeski para la generación de copias y distribuciones.

RPM

Sistema de agrupación de software que permite su fácil instalación en sistemas derivados de la distribución Fedora o Red Hat.

Script

Fichero de código interpretado en el lenguaje del shell de Linux. Son ficheros de texto sin formato.

SUSE

SUSE Linux es la empresa adquirida por Novell (y esta por Attachmate) que oferta su distribución bajo el mismo nombre. Como distribución es una de las más conocidas a nivel mundial y tiene una versión de código abierto llamada openSUSE. La versión original de SUSE proviene de una empresa alemana como una derivación de Slackware. La distribución es especialmente reconocida gracias a su gestor de configuración YaST.

SUSE Studio

Web de la empresa SUSE que permite crear una distribución personalizada.

Ubuntu

Es la distribución de GNU/Linux de mayor difusión, basada en Debian. Es un proyecto que nació gracias a la iniciativa personal del magnate Mark Shuttleworth para crear un sistema operativo gratuito y accesible a todo el mundo.

Esta distribución tiene soporte específico para empresas mediante la empresa Canonical Ltd. y se edita en versiones regulares cada 6 meses y en versiones de larga duración *LTS* cada dos años.

4 Bibliografía y referencias

Sobre *Ubuntu*

http://www.ubuntu.com

http://www.canonical.com

https://help.ubuntu.com/community/LiveCDCustomization

Sobre *LFS*

http://www.linuxfromscratch.org/index.html

http://tldp.org/

http://www.tldp.org/HOWTO/Software-Building-HOWTO.html

http://www.culmination.org/Mike/2.6-udev-nptl-bootcd.txt

Sobre herramientas *GNU/Linux*

http://www.srbyte.com/2010/06/personalizando-mi-distro-linux-antes-de.html

http://susestudio.com

https://build.reconstructor.org

http://remastersys.sourceforge.net

http://www.geekconnection.org/remastersys

http://fedoraproject.org/wiki/How_to_create_and_use_a_Live_CD

https://reconstructor.apphosted.com

http://uck.sourceforge.net/

http://www.garfio.org.ar/

http://www.linux-live.org/

http://www.gnewsense.org/Builder?action=show&redirect=Builder%2FBuilder

5 Anexos

5.1 Anexo I – Programa de prueba

Se ha generado una aplicación simple para comprobar el funcionamiento del procedimiento.

Este programa simula la necesidad de proporcionar un software con dependencias no usuales.

Dado que es un entorno gráfico con movimiento es fácil de comprobar si opera con normalidad.

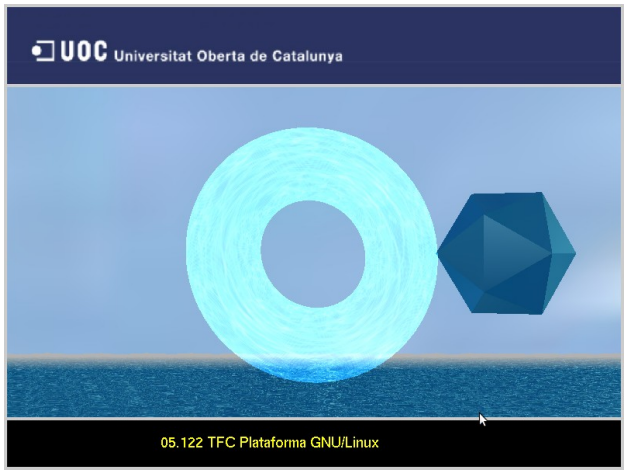

Ilustración 9: Programa de prueba

Esta aplicación tiene la particularidad de usar las librerías gráficas gl, glu y glut. Esto hace que necesite dichas librerías instaladas en el sistema operativo para poder funcionar correctamente. En caso de que el sistema no disponga de esas librerías se muestra un error como en la imagen adjunta.

```
tfc@tfc-desktop:~/Escritorio/Fuentes$ ./tfc-test
./tfc-test: error while loading shared libraries: libglut.so.3: cannot open shar
ed object file: No such file or directory
tfc@tfc-desktop:~/Escritorio/Fuentes$
```

Ilustración 10: Aviso por falta de librerías

También se adjunta el fichero Makefile generado y el fichero 'proyecto tfc test.dbp' para la creación del DEB. Este último contiene la configuración de la herramienta debreate que lo genera http://debreate.sourceforge.net.

La aplicación instalada la conforman los ficheros:

- alpha.ppm – Textura para efectos. Ubicada en /usr/share/applications
- cielo.ppm – Textura para efectos. Ubicada en /usr/share/applications
- logo.ppm – Imagen superior. Ubicada en /usr/share/applications
- reflection.ppm – Textura para efectos. Ubicada en /usr/share/applications
- tfc-test – Aplicación. Ubicada en /usr/share/applications
- TFC_text_1.0.0.desktop - Archivo de configuración de escritorio para el inicio automático de la aplicación con el arranque del sistema operativo. Ubicado en /usr/share/applications.

5.1.1.1 main.cpp

```
#include "ogl_main.h"

#include "ogl_view_titulo.h"
#include "ogl_view_imagen.h"
#include "ogl_view_menu.h"

#include "oTiempo.h"
#include <unistd.h>

#define CD_PathInstall "/usr/local/bin/tfc/"

OTiempo oTiempoVentanaScroll;

int vtnMainId;

float vtnAncho;
float vtnAlto;
int   vtnMarco;
int   vtnPie;

bool vtnMainFullScreen = false;

ST_VTN_DATA vdTitulo;
ST_VTN_DATA vdImagen;
ST_VTN_DATA vdMenu;

int main(int argc, char * argv[])
{
    OGL_Main(argc, argv);
    return 0;
}

void OGL_Main(int argc, char * argv[])
{
       vtnAncho = 800;
       vtnAlto  = 600;
       vtnMarco = 4;
       vtnPie   = 60;
       OGL_Main_Calcula_CoordSubventanas(vtnAncho, vtnAlto);

      //////////////////////////////////////// Parametros generales y creación de la
ventana principal
       glutInit(&argc, argv);
       glutInitDisplayMode( GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE );
       glutInitWindowSize(vtnAncho, vtnAlto);
       glutInitWindowPosition(100, 15);
       vtnMainId = glutCreateWindow("05.122 TFC Plataforma GNU/Linux");

        //////////////////////////////////////// Inicializa funciones de la ventana
principal
       glutKeyboardFunc(  OGL_Main_Keyboard );
       glutReshapeFunc(   OGL_Main_Reshape  );
       glutDisplayFunc(   OGL_Main_Dibuja   );

    //////////////////////////////////////// Crea Subventanas
    OGL_View1_Inicializa();
    OGL_View2_Inicializa();
    OGL_View3_Inicializa();

    //////////////////////////////////////// Fuerza redibujado de las subventanas
    OGL_Main_Redisplay();

    //////////////////////////////////////// Inicializa Timer de la ventana principal
       glutTimerFunc( 40, OGL_Main_Timer, 1 );

    //////////////////////////////////////// Bucle de OpenGL
       glutMainLoop();
}

void OGL_Main_Keyboard(unsigned char key, int x, int y)
{
    switch (key)
    {
```

```
    case 13:
    case 27:
        exit (0);
        break;
    }
}

void OGL_Main_Calcula_CoordSubventanas(int width, int height)
{
       float rel;

    vtnAncho = width;
    vtnAlto  = height;

    vdTitulo.fPosX  = vtnMarco;
       vdTitulo.fPosY  = vtnMarco;
       vdTitulo.fAncho = vtnAncho - 2 * vtnMarco;
       rel = 1024 / vdTitulo.fAncho;
       vdTitulo.fAlto  = 128 / rel;

    vdImagen.fPosX  = vtnMarco;
       vdImagen.fPosY  = vdTitulo.fAlto + 2 * vtnMarco;
       vdImagen.fAncho = vdTitulo.fAncho;
       vdImagen.fAlto  = vtnAlto - vdTitulo.fAlto - 4 * vtnMarco - vtnPie;

       switch ( vdMenu.iEstado )
       {
           case 1: // Abajo
            vdMenu.fPosY  = vtnAlto - vtnPie - vtnMarco;
            break;
           case 2: // Subiendo
            vdMenu.fPosY = vdMenu.fAuxY;
            if ( vdMenu.fPosY <= vdImagen.fPosY )
            {
                vdMenu.fPosY = vdImagen.fPosY;
                vdMenu.iEstado = 3;
            }
            break;
           case 3: // Arriba
            vdMenu.fPosY = vdImagen.fPosY;
            break;
           case 4: // Bajando
            vdMenu.fPosY = vdMenu.fAuxY;
            if ( vdMenu.fPosY >= vtnAlto - vtnPie - vtnMarco )
            {
                vdMenu.fPosY = vtnAlto - vtnPie - vtnMarco;
                vdMenu.iEstado = 1;
            }
            break;
    }
    vdMenu.fPosX  = vtnMarco;
       vdMenu.fAncho = vdTitulo.fAncho;
    vdMenu.fAlto  = vtnAlto - vdMenu.fPosY - vtnMarco;

}

void OGL_Main_Redisplay()
{
    glutSetWindow( vdTitulo.Id );
    glutPositionWindow( vdTitulo.fPosX,  vdTitulo.fPosY);
    glutReshapeWindow(  vdTitulo.fAncho, vdTitulo.fAlto);
    glutPostRedisplay();

    glutSetWindow( vdImagen.Id );
    glutPositionWindow( vdImagen.fPosX,  vdImagen.fPosY);
    glutReshapeWindow(  vdImagen.fAncho, vdImagen.fAlto);
    glutPostRedisplay();

    glutSetWindow( vdMenu.Id );
    glutPositionWindow( vdMenu.fPosX,  vdMenu.fPosY);
    glutReshapeWindow(  vdMenu.fAncho, vdMenu.fAlto);
    glutPostRedisplay();
}

void OGL_Main_Reshape(int width, int height)
{
```

```
        vtnAncho = width;
        vtnAlto  = height;

        ////////////////////////////////// Ajusta ventana principal
    glViewport(0, 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, width, height, 0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();


        ////////////////////////////////// Ajusta subventanas
        OGL_Main_Calcula_CoordSubventanas(width, height);
        OGL_Main_Redisplay();
}

void OGL_Main_Timer( int iParam )
{
    bool bCambio = false;
    if ( oTiempoVentanaScroll.Alarma( 40 ) )
    {
        oTiempoVentanaScroll.Inicializa();

        switch ( vdMenu.iEstado )
        {
            case 1: // Abajo
                vdMenu.fAuxY = vtnAlto - vtnPie - vtnMarco;
                break;
            case 2: // Subiendo
                bCambio = true;
                vdMenu.fAuxY -= 40;
                if ( vdMenu.fAuxY <= vdImagen.fPosY )
                {
                    vdMenu.fAuxY = vdImagen.fPosY;
                    vdMenu.iEstado = 3;
                }
                break;
            case 3: // Arriba
                vdMenu.fAuxY = vdImagen.fPosY;
                break;
            case 4: // Bajando
                bCambio = true;
                vdMenu.fAuxY += 40;
                if ( vdMenu.fAuxY >= vtnAlto - vtnPie - vtnMarco )
                {
                    vdMenu.fAuxY = vtnAlto - vtnPie - vtnMarco;
                    vdMenu.iEstado = 1;
                }
                break;
        }
    }

    if ( bCambio )
    {
        OGL_Main_Calcula_CoordSubventanas(vtnAncho, vtnAlto);
        OGL_Main_Redisplay();
    } else {
        glutSetWindow( vdImagen.Id );
        glutPostRedisplay();

        glutSetWindow( vdMenu.Id );
        glutPostRedisplay();
    }

       glutTimerFunc( 40, OGL_Main_Timer, 1 );
}

void OGL_Main_Dibuja()
{
    glClearColor(0.8, 0.8, 0.8, 0.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glutSwapBuffers();
}
```

5.1.1.2 ogl_texto.cpp

```
#include "ogl_texto.h"

#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>

GLvoid *font_style = GLUT_BITMAP_TIMES_ROMAN_10;

void OGL_Texto_Fuente(char* name, int size)
{
    font_style = GLUT_BITMAP_HELVETICA_10;
    if (strcmp(name, "helvetica") == 0)
    {
        if (size == 12) font_style = GLUT_BITMAP_HELVETICA_12;
        else if (size == 18)
            font_style = GLUT_BITMAP_HELVETICA_18;
    } else if (strcmp(name, "times roman") == 0) {
        font_style = GLUT_BITMAP_TIMES_ROMAN_10;
        if (size == 24)
            font_style = GLUT_BITMAP_TIMES_ROMAN_24;
    } else if (strcmp(name, "8x13") == 0) {
        font_style = GLUT_BITMAP_8_BY_13;
    } else if (strcmp(name, "9x15") == 0) {
        font_style = GLUT_BITMAP_9_BY_15;
    }
}

int OGL_Texto_ancho(char* format, ...)
{
    int iAncho = 0;
    va_list args;
    char buffer[1024], *s;

    va_start(args, format);
    vsprintf(buffer, format, args);
    va_end(args);

    for (s = buffer; *s; s++)
        iAncho += glutBitmapWidth( font_style, *s);

    return iAncho;
}

void OGL_Texto(int x, int y, char* format, ...)
{
    va_list args;
    char buffer[1024], *s;

    va_start(args, format);
    vsprintf(buffer, format, args);
    va_end(args);

    glRasterPos2i(x, y);
    for (s = buffer; *s; s++)
        glutBitmapCharacter(font_style, *s);
}
```

5.1.1.3 ogl_textura.cpp

```
#include "ogl_textura.h"

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

GLenum minfilter = GL_NEAREST;
GLenum magfilter = GL_NEAREST;
```

```
GLenum env = GL_MODULATE;

GLenum wraps = GL_REPEAT;
GLenum wrapt = GL_REPEAT;

cell bcolor[4] = {
    { 39, 240, 30, 0.0, 1.0, 1.0, 0.01,
        "Specifies red component of texture border color.", "%.2f" },
    { 40, 300, 30, 0.0, 1.0, 0.0, 0.01,
    "Specifies green component of texture border color.", "%.2f" },
    { 41, 360, 30, 0.0, 1.0, 0.0, 0.01,
    "Specifies blue component of texture border color.", "%.2f" },
    { 42, 420, 30, 0.0, 1.0, 1.0, 0.01,
    "Specifies alpha component of texture border color.", "%.2f" },
};

cell ecolor[4] = {
    { 35, 240, 60, 0.0, 1.0, 0.0, 0.01,
        "Specifies red component of texture environment color.", "%.2f" },
    { 36, 300, 60, 0.0, 1.0, 1.0, 0.01,
    "Specifies green component of texture environment color.", "%.2f" },
    { 37, 360, 60, 0.0, 1.0, 0.0, 0.01,
    "Specifies blue component of texture environment color.", "%.2f" },
    { 38, 420, 60, 0.0, 1.0, 1.0, 0.01,
    "Specifies alpha component of texture environment color.", "%.2f" },
};

void cell_vector(float* dst, cell* cell, int num)
{
    while (--num >= 0)
        dst[num] = cell[num].value;
}

GLubyte* OGL_Tex_LeePPM(char* filename, int* width, int* height)
{
    FILE* fp;
    int i, w, h, d;
    unsigned char* image;
    char head[70];          /* max line <= 70 in PPM (per spec). */
    char cFilePath[1024];
    strcpy( cFilePath, CD_PathInstall );
    strcat( cFilePath, filename );

    fp = fopen(cFilePath, "rb");
    if (!fp) {
        perror(filename);
        return NULL;
    }

    /* grab first two chars of the file and make sure that it has the
       correct magic cookie for a raw PPM file. */
    fgets(head, 70, fp);
    if (strncmp(head, "P6", 2)) {
        fprintf(stderr, "%s: Not a raw PPM file\n", filename);
        return NULL;
    }

    /* grab the three elements in the header (width, height, maxval). */
    i = 0;
    while(i < 3) {
        fgets(head, 70, fp);
        if (head[0] == '#')     /* skip comments. */
            continue;
        if (i == 0)
            i += sscanf(head, "%d %d %d", &w, &h, &d);
        else if (i == 1)
            i += sscanf(head, "%d %d", &h, &d);
        else if (i == 2)
            i += sscanf(head, "%d", &d);
    }

    /* grab all the image data in one fell swoop. */
    image = (unsigned char*)malloc(sizeof(unsigned char)*w*h*3);
    fread(image, sizeof(unsigned char), w*h*3, fp);
    fclose(fp);
```

```
    *width = w;
    *height = h;
    return image;
}
```

5.1.1.4 ogl_view_imagen.cpp

```
#include "ogl_view_imagen.h"
#include "ogl_main.h"
#include "ogl_textura.h"
#include "oTiempo.h"

#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

#define        RESOLUTION 64

OTiempo oTiempoImagen;

GLubyte* imgImagen = 0;
GLubyte* imgAlpha = 0;
GLubyte* imgReflection = 0;

GLuint  textureAlpha;
GLuint  textureReflection;
GLuint  textureTotal;
GLuint  textureCielo;
unsigned char total_texture[4 * 256 * 256];
float fImagenCieloX = 0;
float fImagenCieloY = 0;
GLUnurbsObj * objAguaNurb;

void OGL_View2_Dibuja();
void OGL_View2_Reshape(int width, int height);

GLfloat light_ambient [] = {.4, .4, .4, 1.0};
GLfloat light_diffuse [] = {1, 1, 1, 1.0};
GLfloat light_specular [] = {1.0, 1.0, 1.0, 1.0};
GLfloat light_position [] = {0.0, 4.0, 15.0, 0.0};
GLfloat light_direction [] = {0.0, 0.0, -1.0};
GLfloat light_spot_cutoff [] = {45.0, 45.0, 45.0};
GLfloat light_spot_exponent [] = {0.0, 0.0, 0.0};

GLfloat mat_ambient[] = {0.25, 0.25, 0.25, 1}; // Vidrio
GLfloat mat_diffuse[] = {0.40, 0.40, 0.40, 1}; // Vidrio
GLfloat mat_specular[] = {0.774597, 0.774597, 0.774597, 1};
GLfloat mat_shininess[] = {76.8};

GLfloat mat_suelo_ambient[] = {0.679688, 0.5625, 0.4375, 1};
GLfloat mat_suelo_diffuse[] = {0.679688, 0.5625, 0.4375, 1};
GLfloat mat_suelo_specular[] = {0.87125, 0.78125, 0.78125, 1};
GLfloat mat_suelo_shininess[] = {39.68};
GLfloat mat_suelo[] = {0.9, 0.9, 0.9, 39.68}; // mate

GLfloat mat_cielo_ambient[] = {0.25, 0.25, 0.25, 1}; // Vidrio
GLfloat mat_cielo_diffuse[] = {0.40, 0.40, 0.40, 1}; // Vidrio
GLfloat mat_cielo_specular[] = {0.774597, 0.774597, 0.774597, 1};
GLfloat mat_cielo_shininess[] = {76.8};

GLfloat mat_goma_ambient[] = {0.02, 0.02, 0.02, 1};
GLfloat mat_goma_diffuse[] = {0.01, 0.01, 0.01, 1};
GLfloat mat_goma_specular[] = {0.4, 0.4, 0.4, 1};
GLfloat mat_goma_shininess[] = {10};

static float   surface[6 * RESOLUTION * (RESOLUTION + 1)];
static float   normal[6 * RESOLUTION * (RESOLUTION + 1)];
static int     wire_frame = 0;
static int     normals = 0;

static float   z (const float x, const float y, const float t)
{
  const float x2 = x - 30;
```

```
  const float y2 = y + 10;
  const float xx = x2 * x2;
  const float yy = y2 * y2;
  return ((2 * sinf (20 * sqrtf (xx + yy) - 4 * t + .1 * drand48()) ) / 200);//+Noise
(10 * x, 10 * y, t, 0)) / 200);
}

void OGL_View2_Inicializa()
{
       vdImagen.Id  =  glutCreateSubWindow(  vtnMainId,  vdImagen.fPosX,  vdImagen.fPosY,
vdImagen.fAncho, vdImagen.fAlto );
    glutReshapeFunc(  OGL_View2_Reshape );
    glutDisplayFunc(  OGL_View2_Dibuja  );
    glutKeyboardFunc( OGL_Main_Keyboard );

    if ( imgImagen ) free( imgImagen );
    int iWidth, iHeight;

    glGenTextures (1, &textureAlpha);
    imgAlpha = OGL_Tex_LeePPM( (char *) "alpha.ppm", &iWidth, &iHeight);
    if ( !imgAlpha ) exit(2);

    glGenTextures (1, &textureReflection);
    imgReflection = OGL_Tex_LeePPM( (char *) "reflection.ppm", &iWidth, &iHeight);
    if ( !imgReflection ) exit(2);

    glGenTextures (1, &textureTotal);
    for (int i = 0; i < 256 * 256; i++)
    {
        total_texture[4 * i] = imgReflection[3 * i];
        total_texture[4 * i + 1] = imgReflection[3 * i + 1];
        total_texture[4 * i + 2] = imgReflection[3 * i + 2];
        total_texture[4 * i + 3] = imgAlpha[i];
    }
    glBindTexture (GL_TEXTURE_2D, textureTotal);
      gluBuild2DMipmaps  (GL_TEXTURE_2D,  GL_RGBA,  256,  256,  GL_RGBA,  GL_UNSIGNED_BYTE,
total_texture);

    ////////////////////////////////////////////////////////////////////////
    glGenTextures (1, &textureCielo);
    imgImagen = OGL_Tex_LeePPM( (char *) "cielo.ppm", &iWidth, &iHeight);
    if ( !imgImagen ) exit(2);

    GLfloat env_color[4], border_color[4];

    cell_vector(env_color, ecolor, 4);
    cell_vector(border_color, bcolor, 4);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, minfilter);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, magfilter);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE); // GL_REPLACE GL_BLEND
    glTexEnvfv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_COLOR, env_color);
    glTexParameterfv(GL_TEXTURE_2D, GL_TEXTURE_BORDER_COLOR, border_color);
    glBindTexture (GL_TEXTURE_2D, textureCielo);
       gluBuild2DMipmaps(GL_TEXTURE_2D,  3,  iWidth,  iHeight,  GL_RGB,  GL_UNSIGNED_BYTE,
imgImagen);


    glMatrixMode(GL_TEXTURE);
    glScalef(20,20,1);
}

void OGL_View2_Reshape(int width, int height)
{
    glViewport(0, 0, width, height);

    ////////////////////////////////// Inicializa Vista
    glMatrixMode( GL_MODELVIEW_MATRIX );
    glLoadIdentity();

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(15, vdImagen.fAncho / vdImagen.fAlto, 1, 1000.0);
    glMatrixMode(GL_MODELVIEW);
```

```
    glLoadIdentity();
    gluLookAt(0.0, 0.0, 1.0,  0.0, 1000.0, 2.0, 0.0, 0.0, 1.0);

    glEnable(GL_LIGHT0);
    glEnable(GL_COLOR_MATERIAL);

    ////////////////////////////////// Niebla
    float fog_color[] = {0.7, 0.7, 0.7, 1};
    glEnable(GL_FOG);
    glFogi(GL_FOG_MODE, GL_LINEAR);
    glFogf(GL_FOG_START, 15);
    glFogf(GL_FOG_END, 17);
    glFogfv(GL_FOG_COLOR, fog_color);


    ////////////////////////////////// Nurbs
   objAguaNurb = gluNewNurbsRenderer();
   gluNurbsProperty(objAguaNurb, GLU_DISPLAY_MODE, GLU_FILL);
}

void Evaluador( int iMax, int jMax )
{
    int i, j;
   for (j = 0; j <= jMax; j++) {
      glBegin(GL_LINE_STRIP);
      for (i = 0; i <= iMax; i++)
         glEvalCoord2f((GLfloat)i/iMax, (GLfloat)j/jMax);
      glEnd();
      glBegin(GL_LINE_STRIP);
      for (i = 0; i <= iMax; i++)
         glEvalCoord2f((GLfloat)j/jMax, (GLfloat)i/iMax);
      glEnd();
   }
}

void nurbsError(GLenum errorCode)
{
   const GLubyte *estring;

   estring = gluErrorString(errorCode);
   fprintf (stderr, "Nurbs Error: %s\n", estring);
   exit (0);
}
void Agua();
void Agua2();

void Agua3();
void Agua4();
void Agua5();
void Agua6();

void OGL_View2_Dibuja()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glLightfv (GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv (GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv (GL_LIGHT0, GL_SPECULAR, light_specular);
    glLightfv (GL_LIGHT0, GL_POSITION, light_position);
    glLightfv (GL_LIGHT0, GL_SPOT_DIRECTION, light_direction);
    glLightfv (GL_LIGHT0, GL_SPOT_CUTOFF, light_spot_cutoff);
    glLightfv (GL_LIGHT0, GL_SPOT_EXPONENT, light_spot_exponent);

    ////////////////////////////////////////////////////////////////// Cielo
    glBindTexture (GL_TEXTURE_2D, textureCielo);

    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT); // GL_CLAMP GL_REPEAT
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT); // GL_CLAMP GL_REPEAT
    glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE); // GL_REPLACE GL_BLEND

    glNormal3f(0,0,-1);
       glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, mat_cielo_ambient);
       glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, mat_cielo_diffuse);
       glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mat_cielo_specular);
```

```
      glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, mat_cielo_shininess);

   glPushMatrix();
   glEnable(GL_TEXTURE_2D);
   glEnable(GL_LIGHTING);
   glColor3f (0.0, 0.0, 1);
   glBegin(GL_QUADS);
   glTexCoord2f (0,0); glVertex3f(-15, 3, 2);
   glTexCoord2f (1,0); glVertex3f(-15,  17, -3.5);
   glTexCoord2f (1,1); glVertex3f( 15,  17, -3.5);
   glTexCoord2f (0,1); glVertex3f( 15, 3, 2);
   glEnd();
      glDisable( GL_BLEND );
   glDisable(GL_TEXTURE_2D);

   glPopMatrix();

   ////////////////////////////////////////////////////////////// Agua

   glPushMatrix();
   Agua6();
   glPopMatrix();

   ////////////////////////////////////////////////////////////// Icosaedro
   static float fFiguraAngulo = 0;
   static float fFiguraRotacion = 0;
   fFiguraAngulo += 2;
   if ( fFiguraAngulo > 360 ) fFiguraAngulo = 0;
   fFiguraRotacion += 5;
   if ( fFiguraRotacion > 360 ) fFiguraRotacion = 0;

   glNormal3f(0,0,1);
   glPushMatrix();
   glScalef(0.5,0.5,0.5);
   glTranslatef(0,17,2);
   glRotatef(-fFiguraAngulo, 0, 0, 1);
   glTranslatef(2.8,0,0);
   glRotatef(fFiguraRotacion, 0, 0, 1);

   glDisable(GL_BLEND);
   glColor3f(0.1, 0.1, 0.1);
   glEnable(GL_COLOR_MATERIAL);
      glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT,    mat_goma_ambient);
      glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE,    mat_goma_diffuse);
      glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR,   mat_goma_specular);
      glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS,  mat_goma_shininess);

   glutSolidIcosahedron();
      glPopMatrix();

   ////////////////////////////////////////////////////////////// Toroide
   glNormal3f(0,0,1);

   glPushMatrix();
   glScalef(0.5,0.5,0.5);
   glTranslatef(0,17,2);
   static float fToroideAngulo = 0;
   if ( oTiempoImagen.Alarma(40) )
   {
       oTiempoImagen.Inicializa();
       fToroideAngulo += 2;
       if ( fToroideAngulo > 360 ) fToroideAngulo = 0;
   }

   glEnable(GL_BLEND);
   glBlendFunc(GL_SRC_ALPHA, GL_ONE);

   glRotatef(fToroideAngulo, 0.3, 0.7, 0.5);
   glColor4f(0.2, 0.2, 0.9, 0.4);
      glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, mat_ambient);
      glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, mat_diffuse);
      glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mat_specular);
      glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, mat_shininess);
      glutSolidTorus(0.5, 1.25, 64, 64);

      glBlendFunc(GL_ONE, GL_ZERO);
```

```
    glDisable(GL_BLEND);
       glPopMatrix();

    glutSwapBuffers();
}

void Agua6()
{
    glTranslatef (-4, 8.5, -0.4);

    const float t = glutGet (GLUT_ELAPSED_TIME) / 1000.;

    const float delta = 10. / RESOLUTION;
    const unsigned int length = 2 * (RESOLUTION + 1);
    const float xn = (RESOLUTION + 1) * delta + 1;
    unsigned int i;
    unsigned int j;
    float x;
    float y;
    unsigned int indice;
    unsigned int preindice;

    // Yes, I know, this is quite ugly...
    float v1x;
    float v1y;
    float v1z;

    float v2x;
    float v2y;
    float v2z;

    float v3x;
    float v3y;
    float v3z;

    float vax;
    float vay;
    float vaz;

    float vbx;
    float vby;
    float vbz;

    float nx;
    float ny;
    float nz;

    float l;

    glRotatef (90, 1, 0, 0);
    glRotatef (90, 0, 1, 0);

  // Vertices
  for (j = 0; j < RESOLUTION; j++)
    {
      y = (j + 1) * delta - 1;
      for (i = 0; i <= RESOLUTION; i++)
       {
         indice = 6 * (i + j * (RESOLUTION + 1));

         x = i * delta - 1;
         surface[indice + 3] = x;
         surface[indice + 4] = z (x, y, t);
         surface[indice + 5] = y;
         if (j != 0)
           {
             // Values were computed during the previous loop
             preindice = 6 * (i + (j - 1) * (RESOLUTION + 1));
             surface[indice] = surface[preindice + 3];
             surface[indice + 1] = surface[preindice + 4];
             surface[indice + 2] = surface[preindice + 5];
           }
         else
           {
             surface[indice] = x;
             surface[indice + 1] = z (x, -1, t);
```

```
              surface[indice + 2] = -1;
            }
        }
    }

  // Normals
  for (j = 0; j < RESOLUTION; j++)
    for (i = 0; i <= RESOLUTION; i++)
      {
       indice = 6 * (i + j * (RESOLUTION + 1));

       v1x = surface[indice + 3];
       v1y = surface[indice + 4];
       v1z = surface[indice + 5];

       v2x = v1x;
       v2y = surface[indice + 1];
       v2z = surface[indice + 2];

       if (i < RESOLUTION)
         {
           v3x = surface[indice + 9];
           v3y = surface[indice + 10];
           v3z = v1z;
         }
       else
         {
           v3x = xn;
           v3y = z (xn, v1z, t);
           v3z = v1z;
         }

       vax =  v2x - v1x;
       vay =  v2y - v1y;
       vaz =  v2z - v1z;

       vbx = v3x - v1x;
       vby = v3y - v1y;
       vbz = v3z - v1z;

       nx = (vby * vaz) - (vbz * vay);
       ny = (vbz * vax) - (vbx * vaz);
       nz = (vbx * vay) - (vby * vax);

       l = sqrtf (nx * nx + ny * ny + nz * nz);
       if (l != 0)
         {
           l = 1 / l;
           normal[indice + 3] = nx * l;
           normal[indice + 4] = ny * l;
           normal[indice + 5] = nz * l;
         }
       else
         {
           normal[indice + 3] = 0;
           normal[indice + 4] = 1;
           normal[indice + 5] = 0;
         }


       if (j != 0)
         {
           // Values were computed during the previous loop
           preindice = 6 * (i + (j - 1) * (RESOLUTION + 1));
           normal[indice] = normal[preindice + 3];
           normal[indice + 1] = normal[preindice + 4];
           normal[indice + 2] = normal[preindice + 5];
         }
       else
         {
//         v1x = v1x;
           v1y = z (v1x, (j - 1) * delta - 1, t);
           v1z = (j - 1) * delta - 1;

//         v3x = v3x;
           v3y = z (v3x, v2z, t);
```

```
            v3z = v2z;

            vax = v1x - v2x;
            vay = v1y - v2y;
            vaz = v1z - v2z;

            vbx = v3x - v2x;
            vby = v3y - v2y;
            vbz = v3z - v2z;

            nx = (vby * vaz) - (vbz * vay);
            ny = (vbz * vax) - (vbx * vaz);
            nz = (vbx * vay) - (vby * vax);

            l = sqrtf (nx * nx + ny * ny + nz * nz);
            if (l != 0)
              {
               l = 1 / l;
               normal[indice] = nx * l;
               normal[indice + 1] = ny * l;
               normal[indice + 2] = nz * l;
              }
            else
              {
               normal[indice] = 0;
               normal[indice + 1] = 1;
               normal[indice + 2] = 0;
              }
          }
      }


  // Render wireframe?
  if (wire_frame != 0)
    glPolygonMode (GL_FRONT_AND_BACK, GL_LINE);

  // The water
    glBindTexture (GL_TEXTURE_2D, textureTotal);
    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
    glEnable (GL_TEXTURE_GEN_S);
    glEnable (GL_TEXTURE_GEN_T);
    glTexGeni (GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
    glTexGeni (GL_T, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);


  glEnable (GL_TEXTURE_2D);
  glColor3f (1, 1, 1);
  glEnableClientState (GL_NORMAL_ARRAY);
  glEnableClientState (GL_VERTEX_ARRAY);
  glNormalPointer (GL_FLOAT, 0, normal);
  glVertexPointer (3, GL_FLOAT, 0, surface);
  for (i = 0; i < RESOLUTION; i++)
    glDrawArrays (GL_TRIANGLE_STRIP, i * length, length);

  // Draw normals?
  if (normals != 0)
    {
      glDisable (GL_TEXTURE_2D);
      glColor3f (1, 0, 0);
      glBegin (GL_LINES);
      for (j = 0; j < RESOLUTION; j++)
       for (i = 0; i <= RESOLUTION; i++)
         {
           indice = 6 * (i + j * (RESOLUTION + 1));
           glVertex3fv (&(surface[indice]));
           glVertex3f (surface[indice] + normal[indice] / 50,
                       surface[indice + 1] + normal[indice + 1] / 50,
                       surface[indice + 2] + normal[indice + 2] / 50);
         }

      glEnd ();
    }
}
```

5.1.1.5 ogl_view_menu.cpp

```
#include "ogl_view_menu.h"
#include "ogl_main.h"
#include "ogl_texto.h"

#include "oTiempo.h"

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

OTiempo oTiempoMenu;
bool bMovTextoIzquierda = true;

void OGL_View3_Dibuja();
void OGL_View3_Reshape(int width, int height);

#define CD_MAX_MENU_ENTRIES  20
#define CD_MAX_MENU_LEN     512
int iMenuTxtMax = 0;
char cMenuTxt[CD_MAX_MENU_ENTRIES][CD_MAX_MENU_LEN];
char cMenuCmd[CD_MAX_MENU_ENTRIES][CD_MAX_MENU_LEN];

void OGL_View3_Inicializa()
{
    vdMenu.fAuxX = 10;

          vdMenu.Id  =  glutCreateSubWindow(  vtnMainId,  vdMenu.fPosX,  vdMenu.fPosY,
vdMenu.fAncho, vdMenu.fAlto);
    glutReshapeFunc(  OGL_View3_Reshape );
    glutDisplayFunc(  OGL_View3_Dibuja  );
    glutKeyboardFunc( OGL_Main_Keyboard );
}

void OGL_View3_Reshape(int width, int height)
{
    glViewport(0, 0, width, height);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, width, height, 0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glClearColor(0.0, 0.0, 0.0, 0.0);

    glMatrixMode(GL_MODELVIEW);
}

void OGL_View3_Dibuja()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);


    OGL_Texto_Fuente( (char *) "helvetica", 18 );
    if ( vdMenu.iEstado == 1 )
    {
        glColor3ub(255, 255, 0);
        int iAncho = OGL_Texto_ancho( (char *) "05.122 TFC Plataforma GNU/Linux");
        OGL_Texto( vdMenu.fAuxX, 35, (char *) "05.122 TFC Plataforma GNU/Linux");

        if ( oTiempoMenu.Alarma( 30 ) )
        {
            oTiempoMenu.Inicializa();
            vdMenu.fAuxX = bMovTextoIzquierda ? vdMenu.fAuxX-2 : vdMenu.fAuxX + 2;

            if ( vdMenu.fAuxX < 10 )
                bMovTextoIzquierda = false;

            if ( vdMenu.fAuxX > vdMenu.fAncho - 10 - iAncho )
                bMovTextoIzquierda = true;
        }
    } else {
        ////////////////////////////// Dibuja opciones Menu
        vdMenu.fAuxX = 10;
```

```
        glColor3ub(255, 255, 0);
        OGL_Texto( 10, 40, (char *) "05.122 TFC Plataforma GNU/Linux");

        int iEspacio = vdMenu.fAlto / (iMenuTxtMax+1);
        int iPosY = iEspacio;
        for ( int i=0; i<iMenuTxtMax; i++ )
        {
            glColor3f(0.75, 0.75, 1);
            glBegin(GL_QUADS);
            glNormal3f(0.0, 0.0, 1.0);
            glTexCoord2f (0,0); glVertex3f(20, iPosY+20, 0);
            glTexCoord2f (1,0); glVertex3f(vdMenu.fAncho - 20, iPosY+20, 0);
            glTexCoord2f (1,1); glVertex3f(vdMenu.fAncho - 20, iPosY-20, 0);
            glTexCoord2f (0,1); glVertex3f(20, iPosY-20, 0);
            glEnd();
            glDisable(GL_TEXTURE_2D);

            glColor3f(0, 0, 0); OGL_Texto( 25+1, iPosY+7, cMenuTxt[i] );
            glColor3f(0, 0, 1); OGL_Texto( 25,   iPosY+6,   cMenuTxt[i] );

            iPosY += iEspacio;
        }
    }

    glutSwapBuffers();
}

void OGL_View3_LeeMenuFch()
{
    FILE* fp;

    iMenuTxtMax = 0;

    fp = fopen("menu.txt", "rb");
    if ( !fp ) perror( "menu.txt" );
    else
    {
        bool bFin = false;
        for ( int i=0; !bFin && i<CD_MAX_MENU_ENTRIES; i++ )
        {
            if ( !fgets(cMenuTxt[i], CD_MAX_MENU_LEN-1, fp) ) bFin = true;
            else
                if ( !fgets(cMenuCmd[i], CD_MAX_MENU_LEN-1, fp) ) bFin = true;
                else
                    iMenuTxtMax = i+1;
        }
        fclose(fp);
    }
}
```

5.1.1.6 ogl_view_titulo.cpp

```
#include "ogl_view_titulo.h"
#include "ogl_main.h"
#include "ogl_textura.h"

GLubyte* imgTitulo = 0;

void OGL_View1_Dibuja();
void OGL_View1_Reshape(int width, int height);


void OGL_View1_Inicializa()
{
       vdTitulo.Id  =  glutCreateSubWindow(  vtnMainId,  vdTitulo.fPosX,  vdTitulo.fPosY,
vdTitulo.fAncho, vdTitulo.fAlto);
    glutReshapeFunc(  OGL_View1_Reshape );
    glutDisplayFunc(  OGL_View1_Dibuja  );
    glutKeyboardFunc( OGL_Main_Keyboard );

    if ( imgTitulo ) free( imgTitulo );
    int iWidth, iHeight;
    imgTitulo = OGL_Tex_LeePPM( (char *) "logo.ppm", &iWidth, &iHeight);
    if ( !imgTitulo ) exit(2);
```

```
    GLfloat env_color[4], border_color[4];

    cell_vector(env_color, ecolor, 4);
    cell_vector(border_color, bcolor, 4);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, minfilter);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, magfilter);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, wraps);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, wrapt);
    glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, env);
    glTexEnvfv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_COLOR, env_color);
    glTexParameterfv(GL_TEXTURE_2D, GL_TEXTURE_BORDER_COLOR, border_color);
       gluBuild2DMipmaps(GL_TEXTURE_2D, 3, iWidth, iHeight, GL_RGB, GL_UNSIGNED_BYTE,
imgTitulo);
}

void OGL_View1_Reshape(int width, int height)
{
    glViewport(0, 0, width, height);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, width, height, 0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glClearColor(0.0, 0.0, 0.0, 0.0);

    glMatrixMode(GL_MODELVIEW);
}

void OGL_View1_Dibuja()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glPushMatrix();

    glEnable(GL_TEXTURE_2D);
    glColor3f(1, 1, 1);
    glBegin(GL_QUADS);
    glNormal3f(0.0, 0.0, 1.0);
    glTexCoord2f (0,0); glVertex3f(0, 0, 0);
    glTexCoord2f (1,0); glVertex3f(vdTitulo.fAncho, 0, 0);
    glTexCoord2f (1,1); glVertex3f(vdTitulo.fAncho, vdTitulo.fAlto, 0);
    glTexCoord2f (0,1); glVertex3f(0, vdTitulo.fAlto, 0);
    glEnd();
    glDisable(GL_TEXTURE_2D);

    glPopMatrix();

    glutSwapBuffers();
}
```

5.1.1.7 oTiempo.cpp

```
#include "oTiempo.h"

OTiempo::OTiempo()
{
    Inicializa();
}

void OTiempo::Inicializa()
{
    ftime( &t_inicial );
}

unsigned int OTiempo::Transcurrido()
{
    struct timeb t_actual;
    ftime( &t_actual );

        unsigned int t_transcurrido = (unsigned int) (1000.0 * (t_actual.time -
t_inicial.time) + (t_actual.millitm - t_inicial.millitm));

    return t_transcurrido;
```

```
}

bool OTiempo::Alarma( unsigned int milisegundos)
{
    return Transcurrido() >= milisegundos;
}
```

5.1.1.8 ogl_texto.h

```
#ifndef OGL_TEXTO_H_INCLUDED
#define OGL_TEXTO_H_INCLUDED

void OGL_Texto_Fuente(char* name, int size);
int  OGL_Texto_ancho( char* format, ...);
void OGL_Texto(int x, int y, char * texto, ...);

#endif // OGL_TEXTO_H_INCLUDED
```

5.1.1.9 ogl_textura.h

```
#ifndef OGL_TEXTURA_H_INCLUDED
#define OGL_TEXTURA_H_INCLUDED

#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

extern GLenum minfilter;
extern GLenum magfilter;
extern GLenum env;
extern GLenum wraps;
extern GLenum wrapt;

typedef struct _cell {
    int id;
    int x, y;
    float min, max;
    float value;
    float step;
    char* info;
    char* format;
} cell;

extern cell bcolor[4];
extern cell ecolor[4];

void cell_vector(float* dst, cell* cell, int num);

GLubyte* OGL_Tex_LeePPM(char* filename, int* width, int* height);

#define CD_PathInstall "/usr/local/bin/tfc/"

#endif // OGL_TEXTURA_H_INCLUDED
```

5.1.1.10ogl_view_imagen.h

```
#ifndef OGL_VIEW_IMAGEN_H_INCLUDED
#define OGL_VIEW_IMAGEN_H_INCLUDED

#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

void OGL_View2_Inicializa();

#endif // OGL_VIEW_IMAGEN_H_INCLUDED
```

5.1.1.11 ogl_view_menu.h

```
#ifndef OGL_VIEW_MENU_H_INCLUDED
#define OGL_VIEW_MENU_H_INCLUDED
```

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

void OGL_View3_Inicializa();
void OGL_View3_Mouse(int button, int state, int X, int Y);

#endif // OGL_VIEW_MENU_H_INCLUDED
```

5.1.1.12 ogl_view_titulo.h

```
#ifndef OGL_VIEW_TITULO_H_INCLUDED
#define OGL_VIEW_TITULO_H_INCLUDED

#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

extern int vtnTituloId;

void OGL_View1_Inicializa();

#endif // OGL_VIEW_TITULO_H_INCLUDED
```

5.1.1.13 oTiempo.h

```
#ifndef OTIEMPO_H_INCLUDED
#define OTIEMPO_H_INCLUDED

#include <sys/timeb.h>

class OTiempo
{
    public:
        OTiempo();
        void Inicializa();
        unsigned int Transcurrido();
        bool Alarma( unsigned int milisegundos);

    private:
        struct timeb t_inicial;
};


#endif // OTIEMPO_H_INCLUDED
```

5.1.1.14 ogl_main.h

```
#ifndef OGL_MAIN_H_INCLUDED
#define OGL_MAIN_H_INCLUDED

#include "ogl_main_globaldata.h"

void OGL_Main(int argc, char * argv[]);

void OGL_Main_Reshape(int width, int height);
void OGL_Main_Keyboard(unsigned char key, int x, int y);
void OGL_Main_Timer(int iParam);
void OGL_Main_Dibuja();

void OGL_Main_Calcula_CoordSubventanas(int width, int height);
void OGL_Main_Redisplay();

#endif // OGL_MAIN_H_INCLUDED
```

5.1.1.15 ogl_main_globaldata.h

```
#ifndef OGL_MAIN_GLOBALDATA_H_INCLUDED
#define OGL_MAIN_GLOBALDATA_H_INCLUDED

#include <GL/gl.h>
#include <GL/glu.h>
```

```
#include <GL/glut.h>

extern int vtnMainId;

extern float vtnAncho;
extern float vtnAlto;
extern int vtnMarco;
extern int vtnPie;

typedef struct _ST_VTN_DATA
{
    GLuint Id;

    float fPosX;
    float fPosY;
    float fAncho;
    float fAlto;
      int iEstado; // valor adicional para el desplazamiento (1 abajo, 2 subiendo, 3
arriba, 4 bajando)
    float fAuxX;  // valor adicional para almacenar la coordenada X del texto
        float  fAuxY;    //  valor  adicional  para  almacenar  la  coordenada  Y  en  los
desplazamientos

    _ST_VTN_DATA()
    {
        fPosX = fPosY = fAncho = fAlto = 0;
        iEstado = 1;
        fAuxX   = 0;
        fAuxY   = 0;
    };
} ST_VTN_DATA;

extern ST_VTN_DATA vdTitulo;
extern ST_VTN_DATA vdImagen;
extern ST_VTN_DATA vdMenu;

#endif // OGL_MAIN_GLOBALDATA_H_INCLUDED
```

5.1.1.16 Makefile

```
# Compilacion
.SUFFIXES: .o .c
.c.o:
       $(CC) -c $(CFLAGS) $<


# Macros

CC = gcc
CFLAGS = -lGL -lGLU -lglut -g -O2 -Wwrite-strings
SRC  =  main.cpp  ogl_texto.cpp  ogl_textura.cpp  ogl_view_imagen.cpp  ogl_view_menu.cpp
ogl_view_titulo.cpp   oTiempo.cpp   /   ogl_main.h   ogl_view_titulo.h   ogl_view_imagen.h
ogl_view_menu.h oTiempo.h
OBJ    =    main.o    ogl_texto.o    ogl_textura.o    ogl_view_imagen.o    ogl_view_menu.o
ogl_view_titulo.o oTiempo.o

# Reglas expl□citas

all: $(OBJ)
       $(CC) $(CFLAGS) -o tfc-test $(OBJ)

clean:
       $(RM) $(OBJ) tfc-test

# Reglas impl□citas
ogl_texto.o: ogl_texto.cpp ogl_texto.h
oTiempo.o: oTiempo.cpp oTiempo.h
ogl_textura.o: ogl_textura.cpp ogl_textura.h
ogl_view_menu.o: ogl_view_menu.cpp ogl_view_menu.h ogl_main.h ogl_texto.h oTiempo.h
ogl_view_titulo.o: ogl_view_titulo.cpp ogl_view_titulo.h ogl_main.h ogl_textura.h
ogl_view_imagen.o:   ogl_view_imagen.cpp   ogl_view_imagen.h   ogl_main.h   ogl_textura.h
oTiempo.h
main.o:   main.cpp   ogl_main.h   ogl_view_titulo.h   ogl_view_imagen.h   ogl_view_menu.h
oTiempo.h
```

```
.PHONY : clean

install:
	mkdir -p ~/tfc
	cp tfc-test ~/tfc/tfc-test
```

5.1.1.17proyecto tfc test.dbp

```
[DEBREATE-1.0]
<<MODE>>0<</MODE>>
<<CTRL>>
Package: TFC-Test
Version: 1.0.0
Section: x11
Maintainer: Manuel Espeleta <mespeleta@uoc.edu>
Priority: optional
Architecture: i386
Depends: freeglut3 (>=2.6.0)
Description: tfc-test
 Programa para pruebas UOC TFC GNU/Linux

<</CTRL>>
<<FILES>>
1
/home/tfc/deb/TFCtestautostart.desktop*       ->       TFCtestautostart.desktop       ->
/usr/share/gnome/autostart
/home/tfc/deb/reflection.ppm -> reflection.ppm -> /usr/local/bin/tfc
/home/tfc/deb/alpha.ppm -> alpha.ppm -> /usr/local/bin/tfc
/home/tfc/deb/tfc-test* -> tfc-test -> /usr/local/bin/tfc
/home/tfc/deb/cielo.ppm -> cielo.ppm -> /usr/local/bin/tfc
/home/tfc/deb/logo.ppm -> logo.ppm -> /usr/local/bin/tfc
<</FILES>>
<<SCRIPTS>>
<<PREINST>>
0
<</PREINST>>
<<POSTINST>>
1
#! /bin/bash -e

ln -fs "/usr/bin/tfc-test" "/usr/bin/tfc-test"
<</POSTINST>>
<<PRERM>>
1
#! /bin/bash -e

rm "/usr/bin/tfc-test"
<</PRERM>>
<<POSTRM>>
0
<</POSTRM>>
<</SCRIPTS>>
<<CHANGELOG>>
<<DEST>>DEFAULT<</DEST>>
TFC-Test (1.0.0) ; urgency=low

  * Primera version

 --  <>  Fri, 17 Dec 2010 19:41:13 +0100


<</CHANGELOG>>
<<COPYRIGHT>>
GNU
<</COPYRIGHT>>
<<MENU>>
1
Name=TFC Text 1.0.0
Version=1.0
Exec=/usr/local/bin/tfc/tfc-test
Comment=
Icon=
Type=Application
Terminal=false
StartupNotify=true
```

```
Encoding=UTF-8
Categories=
<</MENU>>
<<BUILD>>
0
1
1
<</BUILD>>
```

5.1.1.18 TFC_text_1.0.0.desktop

```
[Desktop Entry]
Name=TFC Text 1.0.0
Version=1.0
Exec=/usr/local/bin/tfc/tfc-test
Comment=
Icon=
Type=Application
Terminal=false
StartupNotify=true
Encoding=UTF-8
Categories=
```

5.2 Anexo II – Scripts de instalación

5.2.1 Scripts personalización LiveCD Ubuntu

El proceso se divide en tres ficheros 'proceso1', 'proceso2' y 'proceso3' para evitar conflictos con el cambio de entorno, aunque se ejecutan manualmente cada uno imprime en pantalla los pasos a seguir para el siguiente.

Dado que es un proceso a medida se han de revisar los orígenes de datos y cambiar los nombres de los paquetes que se quieran instalar.

5.2.1.1 proceso1

```
#!/bin/bash

mkdir -p ~/CD
cp ubuntu-10.04-desktop-i386.iso ~/CD
cd ~/CD

# Montado y extracción del contenido de la imagen
mkdir -p mnt
sudo mount -o loop ubuntu-10.04-desktop-i386.iso mnt
mkdir -p extract-cd
rsync --exclude=/casper/filesystem.squashfs -a mnt/ extract-cd

# Extracción del entorno de escritorio
sudo unsquashfs mnt/casper/filesystem.squashfs
sudo mv squashfs-root edit

# Copia de los paquetes propios a instalar
sudo cp *deb edit/var/cache/apt/archives
sudo cp proceso2 edit/tmp

# Cambio de entorno
sudo mount --bind /dev/ edit/dev

echo " "
echo continuar con:
echo "  sudo chroot edit"
echo "  /tmp/proceso2"
```

5.2.1.2 proceso2

```
#!/bin/bash

# Preparación del nuevo entorno (se visualizará un nuevo indicador del sistema)
mount -t proc none /proc
mount -t sysfs none /sys
mount -t devpts none /dev/pts
export HOME=/root
export LC_ALL=C

# Pasos iniciales
dbus-uuidgen > /var/lib/dbus/machine-id
dpkg-divert --local --rename --add /sbin/initctl
ln -s /bin/true /sbin/initctl

# Instalación de programas adicionales copiados anteriormente
dpkg -i /var/cache/apt/archives/freeglut3_2.6.0-0ubuntu2_i386.deb
dpkg -i /var/cache/apt/archives/TFC-Test_1.0.0_i386.deb

# Limpieza del nuevo entorno.
aptitude clean
rm -rf /tmp/* ~/.bash_history
rm /var/lib/dbus/machine-id
```

```
rm /sbin/initctl
dpkg-divert --rename --remove /sbin/initctl

# Salida del nuevo entorno.
umount /proc
umount /sys
umount /dev/pts

echo " "
echo continuar con:
echo "  exit"
echo "  ./proceso3"
```

5.2.1.3 proceso3

```
#!/bin/bash

# Proceso3
echo Proceso3
sudo umount edit/dev

# Configuración de la nueva imagen

# Regenerar la lista de paquetes
echo Regenerar la lista de paquetes
chmod +w extract-cd/casper/filesystem.manifest
sudo chroot edit dpkg-query -W --showformat='${Package} ${Version}\n' > extract-
cd/casper/filesystem.manifest
sudo cp extract-cd/casper/filesystem.manifest extract-cd/casper/filesystem.manifest-
desktop
sudo sed -i '/ubiquity/d' extract-cd/casper/filesystem.manifest-desktop
sudo sed -i '/casper/d' extract-cd/casper/filesystem.manifest-desktop

# Crear el nuevo sistema de ficheros
echo Crear el nuevo sistema de ficheros
sudo rm -f extract-cd/casper/filesystem.squashfs
sudo mksquashfs edit extract-cd/casper/filesystem.squashfs

# Personalizar el fichero de identificación del CD (opcional)
# sudo nano extract-cd/README.diskdefines

# Calcular MD5
echo Calcular MD5
cd extract-cd
sudo rm md5sum.txt
find -type f -print0 | sudo xargs -0 md5sum | grep -v isolinux/boot.cat | sudo tee
md5sum.txt

# Generación de la nueva imagen
echo Generación de la nueva imagen
sudo mkisofs -D -r -V "UOC" -cache-inodes -J -l -b isolinux/isolinux.bin -c
isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o custom.iso .
```