EMERGYA
CONSULTORÍA

# CURSO DE CREACIÓN DE DISTRIBUCIONES: GUADALINEX V3

Proyecto: Monk

Cliente: **Sociedad Andaluza para el Desarrollo de la Sociedad de la Información.**

JUNTA DE ANDALUCIA
CONSEJERÍA DE INNOVACIÓN, CIENCIA Y EMPRESA

Código: **022-06v1.1**

Fecha: **06/06/06**

Referencia: **monk**

Avda. Luis Montoto, 105, 4º K · E41007 Sevilla · Tel. +34 954 98 10 53 · Fax +34 954 98 11 79 · www.emergya.com



Autores:

Carlos Parra Caramargo

Juan Jesús Ojeda Croissier

Rafael Martín de Agar Tirado

# ÍNDICE DE CONTENIDO

# 1.- VISIÓN GENERAL.

## 1.1.- INTRODUCCIÓN A LA CREACIÓN DE DISTRIBUCIONES

La creación de distribuciones es algo más complejo y a la vez menos complicado de lo que la gente suele pensar. La complicación la dan la cantidad de factores que intervienen y que hay que tener en cuenta para hacer una verdadera distribución. Una que realmente sirva para algo más que para demostraciones de la *potencia* de *GNU/Linux*.

Mientras que, lo que hace que sea menos complicada de lo se suele pensar es la tecnología que se usa. Una tecnología bastante avanzada y probada que pone al alcance de *casi* cualquiera la capacidad técnica para llevar a cabo tal empresa.

Cuando hablamos de una *verdadera distribución*, nos referimos a una que se pueda usar, que se pueda instalar, que se pueda actualizar, que sea mantenida posteriormente y que esté pensada para algún grupo de usuarios concretos (ciudadano de a pie, alumno de secundaria, profesional del derecho, administrativos de PYMES, etc) y con algún fin concreto.

Pero para que la distro llegue a ese nivel de *utilidad*, necesitaremos un buen diseño previo y un trabajo posterior. Para dicho diseño no necesitaremos ser programadores, expertos en distribuciones, sistemas de paquetería o similares. Al menos en las primeras fases. Lo único que necesitamos es sentido común, despojarnos de nuestras preferencias y prejuicios, y conocer bien a los futuros usuarios. Para el trabajo posterior sí es posible que necesitemos ayuda de algún experto o serlo nosotros mismos.

En resumen, el proceso de creación de una distribución podríamos dividirlo en las siguientes fases:

- Fase 1: **Análisis y diseño**
- Fase 2: **Personalización**
- Fase 3: **Generación y mantenimiento**

## 1.1.1.- ANÁLISIS Y DISEÑO

Para ilustrar esta fase tenemos este diagrama (Figura 1).

Como podemos observar, todo el diseño parte de nuestro grupo de usuarios, de lo que necesitan y de lo que conocen.

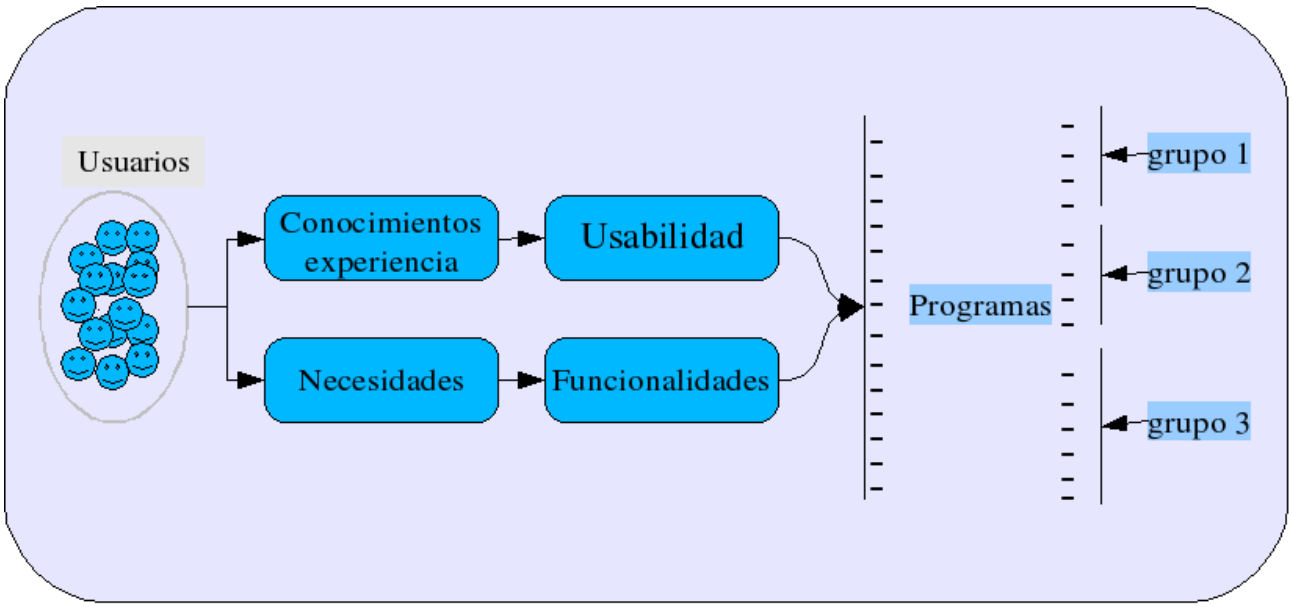



*Figura 1: Diseño orientado al usuario*

Éstos son conceptos muy básicos, pero importantes. Nuestra distro no cumplirá su función si no le permite hacer al usuario lo que necesita hacer. Ni tampoco si el nivel de conocimientos o experiencia es muy superior o inferior al de estos usuarios.

Lo que viene a representar el diagrama (Figura 1) es la secuencia lógica y los elementos implicados en el diseño de la distribución.

De los usuarios obtenemos la información básica: conocimientos/experiencia, necesidades a cubrir.

Esto es importante porque no es lo mismo un usuario avanzado y acostumbrado a poder configurarlo todo según lo que esté haciendo, que alguien con conocimientos ofimáticos básicos, que no tenga tiempo ni ganas de aprender cómo usar y configurar nuevos programas.

Así como tampoco es lo mismo lo que puede necesitar un alumno de primaria que un estudiante universitario o un abogado.

Una vez conocemos las funcionalidades que necesitan cubrir nuestros usuarios ya podemos buscar aplicaciones que lo hagan. Y para organizarlas mejor deberíamos agruparlas por características comunes.

Por ejemplo, un arquitecto necesitará hacer, ver e imprimir planos. Necesitará hacer algún renderizado 3D, leer y mandar correo, consultar información en Internet, redactar informes y presupuestos, gestionar sus proyectos, mantener sus citas y contactos sincronizados con su móvil y/o PDA. (Esto es sólo un análisis simplista y de ejemplo)

Ahora que sabemos esto ya podemos buscar algo que pueda cubrir sus necesidades. Podría ser algo como:

- **Diseño de planos**: QCad
- **3D**: Blender
- **Consultar información en Internet**: Firefox
- **Informes y presupuestos**: OpenOffice
- **Gestión de proyectos**: Planner
- **Correo, citas y contactos**: Evolution

Ahora toca agrupar:

- **Diseño CAD**: QCad y Blender
- **Internet**: Firefox y Evolution
- **Ofimática**: OpenOffice, Planner y Evolution

Y con esto terminaríamos esta primera fase de diseño.

## 1.1.2.- PERSONALIZACIÓN

Aunque el objetivo principal de una distribución es el usuario, intervienen más actores en el proceso de diseño y creación. Por un lado los desarrolladores, por otro los usuarios y, por último, la organización para quien se crea. Ésta puede ser desde una universidad a un departamento de algún organismo público, o incluso una empresa privada.

Pero lo importante es que éstas suelen tener algunos elementos que afectarán al diseño o configuraciones de la distribución como son:

- Imagen corporativa
- Configuraciones físicas de la organización (topología de red, tipo de hardware, servicios de red corporativos, etc)

Todo esto se traduce en configuraciones en las aplicaciones. Así que el siguiente paso es instalar una *distro base*[1], instalar las aplicaciones que previamente seleccionamos y aplicarle los cambios necesarios en sus configuraciones.

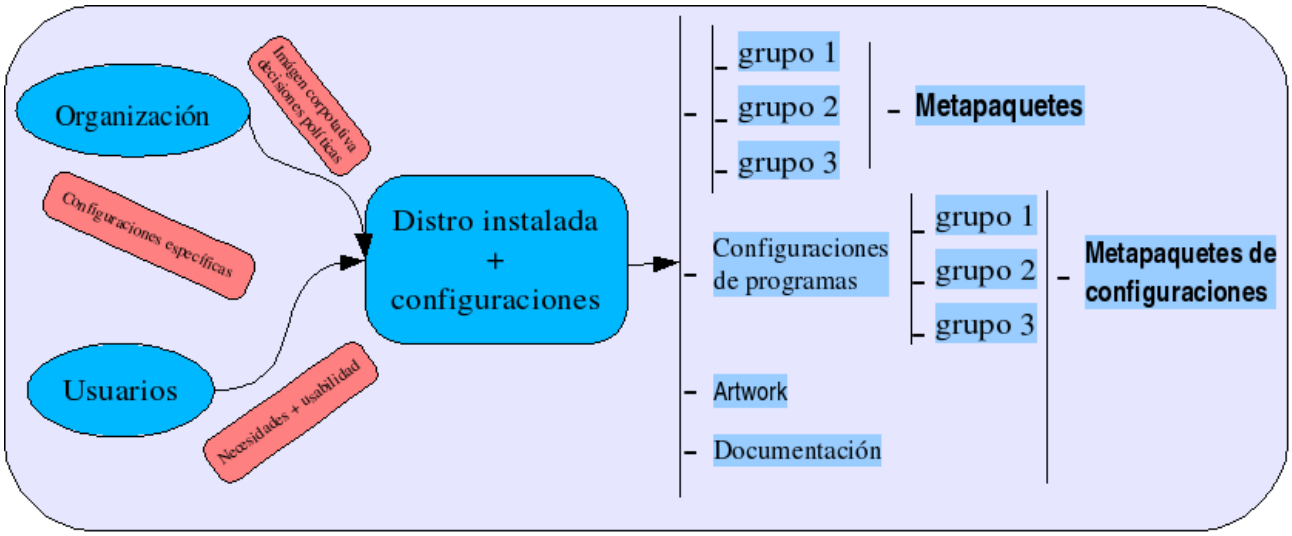



*Figura 2: Aplicar y extraer configuraciones*

A partir de aquí, durante esta fase, sólo nos quedará extraer esas configuraciones y agruparlas tal y como hicimos con las aplicaciones.

Por un lado tendríamos los grupos de aplicaciones, que convertiremos luego en *metapaquetes*[2] y por otro

1 Una distro mínima sobre la que instalar lo que necesitemos

2 Ver capítulo 1.2.1

tenemos las configuraciones de dichas aplicaciones, también agrupadas. Estos grupos darán lugar después a los *metapaquetes de configuración*.

Por otra parte, obtendremos también todo el material necesario para adaptar, tanto la distro como los CDs en los que se distribuirá, a la *imagen corporativa* de la organización.

### 1.1.3.- GENERACIÓN Y MANTENIMIENTO

Esta fase es otra de las que se suele olvidar pero que marca la diferencia entre un *CD live* y una *distribución*.

Para hacer una distribución necesitas un objetivo, un diseño, unas configuraciones predeterminadas y algún medio de *distribución* de la distro. Y de esto último trata esta fase.

Un CD live es un medio óptimo para distribuir nuestra distro por razones de sobra conocidas, pero no podemos quedarnos ahí.

¿Qué pasa si hay fallos de seguridad y hace falta actualizar el "software" afectado? ¿O si sacamos una nueva versión de la distro?

¿Haremos que los usuarios se la reinstalen? ¿No tendría más sentido poder ofrecer unos repositorios de paquetes desde los que poder hacer las actualizaciones oportunas?

De ahí la importancia de guardar configuraciones, grupos de aplicaciones a instalar y cualquier otra diferencia con la *distro padre*, en paquetes. Y después generar nuestros repositorios. Así podremos controlar y mantener de forma eficiente nuestra distribución, usando las herramientas que nos facilita el sistema de paquetería, desde un sólo sitio: el repositorio.

Una vez tengamos el repositorio creado y listo, sólo nos quedará generar los CDs para facilitar la distribución de nuestro *sistema operativo*.

En Guadalinex v3 se han contemplado 2 posibles sistemas de distribución: un instalador no-live, basado en paquetes y un CD live, con su propio instalador. Cada uno convenientemente adaptado a nuestra imagen corporativa ("Artwork").

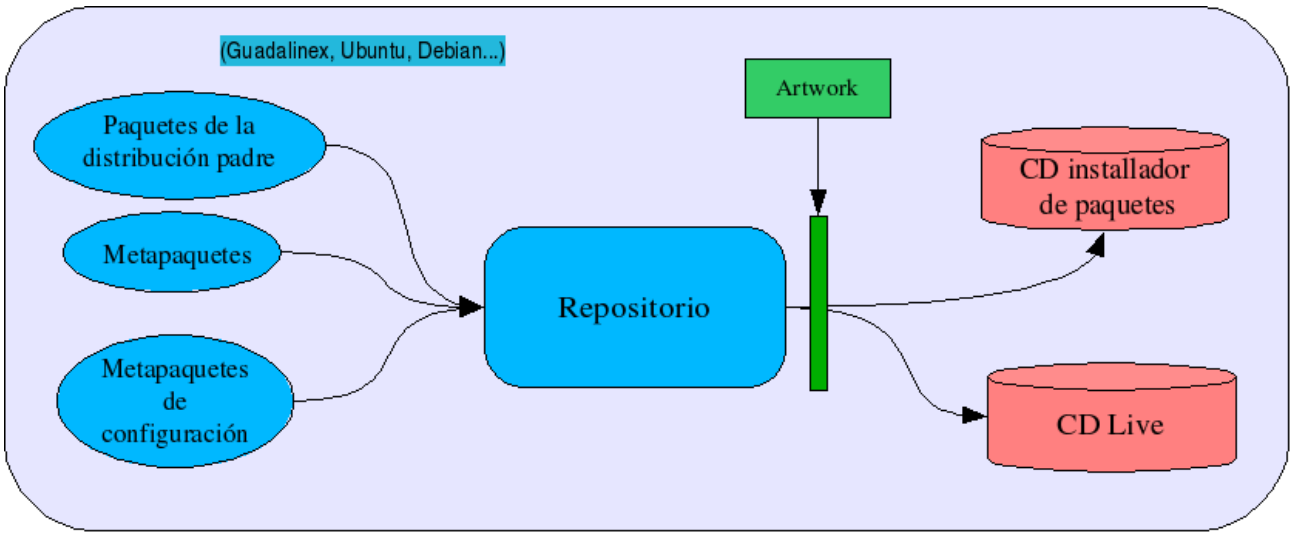



*Figura 3: Crear un repositorio desde donde instalar o generar los CDs distribuibles*

Con esto se termina el diseño y creación de nuestra distro, pero aún nos faltaría algo importante como es el llevar un seguimiento a la misma (solución de bugs, actualizaciones de seguridad, etc) y mantenimiento de los repositorios.

## 1.2.- PERSONALIZACIÓN DE LA DISTRIBUCIÓN.

La personalización de una distribución es un proceso mediante el cual, a partir de una distribución base, se obtiene una nueva adaptada y configurada para unas necesidades concretas previamente definidas. La adaptación y configuración son procesos que se refieren a la obtención de una distribución final con las características software deseadas, lo que incluye:

- Un conjunto de aplicaciones determinado, seleccionado para satisfacer la demanda del conjunto final de usuarios para el que se destina la distribución.

- Configuración adecuada de las mismas para funcionar de la forma que se desea: idioma castellano, directorio donde se almacenan los documentos, y un largo etcétera que depende de las características de cada aplicación.

– Apariencia de los elementos gráficos, donde se incluye la configuración del tema que se va a utilizar en el entorno gráfico, selección de iconos, fondo de escritorio, imagen que se muestra durante el arranque del sistema,...

Una distribución se compone fundamentalmente de elementos software organizados en paquetes, cada uno de los cuales aporta una funcionalidad particular a la distribución.

Es posible personalizar prácticamente todos los elementos de la distribución configurando adecuadamente los paquetes que la componen. Sin embargo, existe un pequeño conjunto de elementos cuya configuración no se limita a lo anterior, siendo ésta un poco diferente. En estos casos, es necesario personalizar el sistema de generación. Por poner un ejemplo, éste es el caso de "isolinux", que se encarga del arranque del sistema para comenzar la instalación. Con isolinux se pueden configurar diferentes opciones de arranque o la imagen inicial que se muestra de la distribución. Estos elementos se configuran a partir del sistema de generación y no configurando los paquetes.

Pero exceptuando estos pocos casos, la gran mayoría de elementos de una distribución se configuran a partir de los paquetes que los componen.

Por este motivo, es necesario seleccionar una buena estrategia para llevar a cabo la configuración de los paquetes.

Posibilidades hay muchas, cada una con sus ventajas e inconvenientes. Por citar algunos ejemplos, se tiene:

– Copia directa de las configuraciones de los paquetes: Este método consiste en preparar todos los ficheros de configuración de los diferentes paquetes y copiarlos en las ubicaciones correspondientes para que surtan efecto las configuraciones. Esto se puede hacer, por ejemplo, creando un fichero "tar.gz" con todas las configuraciones y descomprimiéndolo en el directorio raíz ("/") del sistema que se desea personalizar. Este método es muy sencillo y rápido de llevar a la práctica. Cuenta con los inconvenientes de que es difícil

controlar los cambios que se aplican al sistema final además de que los cambios que se están aplicando no quedan recogidos en ningún paquete, por lo que es difícil deshacerlos para restaurar el sistema y dejarlo en el estado original. Además, de esta manera es complicado mantener las versiones "live" y "no live" sincronizadas, puesto que los cambios no se aplican a partir de paquetes. Asimismo, mediante este método no se tienen control sobre las aplicaciones (paquetes) que se instalan, teniendo que llevar a cabo esta tarea de forma manual. Es decir, no hay ninguna forma de instalar todas las aplicaciones que se deseen a partir de la instalación de algún(os) paquete(s). Esta instalación de aplicaciones habría que llevarla a cabo manualmente.

Ésta fue la aproximación que se siguió en Guadalinex 2004, por poner un ejemplo.

- Recompilación de paquetes: Este método consiste en partir del conjunto original de paquetes Debian[1] de la distribución base y recompilarlos para que incluyan directamente la configuración deseada. Ésta es la aproximación que ha seguido Ubuntu con respecto de Debian: se han recompilado para Ubuntu muchos paquetes Debian para adecuarlos a esta nueva distribución. Aunque este método puede ser adecuado en muchos casos, presenta el inconveniente de que cada vez que cambia el paquete original, si se desea aprovechar las mejoras que se hayan hecho, hay que volver a aplicar los cambios sobre el mismo y regenerar el paquete. Por tanto, el mantenimiento se puede complicar si el conjunto de paquetes que hay que mantener es elevado. Además, desde el punto de vista del trabajo colaborativo, esta aproximación es factible definiendo responsables de cada paquete, que se encargarán de mantener sus paquetes actualizados. Esto es factible cuando el número de desarrolladores es elevado. Por otra parte, como en el caso anterior, también habría que llevar a mano la instalación de los paquetes que se deseen de forma manual, paquete por paquete.

- "dpsyco" y CDD: Éste es un método novedoso e innovador que se ha utilizado con éxito en Guadalinex v3. A grandes rasgos, consiste en implementar una nueva capa de paquetes (metapaquetes de configuración) por encima de los paquetes que se desean configurar. Estos metapaquetes de configuración se encargan de

1 En principio, se podría utilizar cualquier otro sistema de gestión de paquetes (rpm, tgz, etc) pero esto queda fuera del objetivo de este manual.

instalar las aplicaciones necesarias y, una vez éstas se han instalado, las configuran como se haya definido guardando una copia de seguridad de la configuración original. Una de las ventajas que presenta esta aproximación es que a partir de la instalación de un único paquete se lleva a cabo la instalación del sistema completo (gracias a como se ha implementado CDD). Esto permite, por ejemplo, convertir un sistema Ubuntu en un sistema Guadalinex v3 simplemente instalando un paquete, algo que no es factible de forma sencilla con los métodos anteriores. Otra ventaja es que mediante este método no hay que modificar ningún paquete original de la distribución base, de manera que las mejoras y cambios que se lleven a cabo sobre alguno de estos paquetes se pueden incorporar en la distribución final sin necesidad de recompilar ningún paquete. De esta manera, se simplifican las tareas de mantenimiento. Con este método, el trabajo colaborativo se simplifica igualmente porque es muy sencillo trabajar con CDD a partir de un repositorio como svn, y no es necesario tener definidos responsables para cada paquete. También es interesante destacar que con este método, es posible separar en *componentes* la distribución final, de manera que con una definición adecuada de los componentes, es posible crear una nueva distribución a base de combinar adecuadamente los componentes que ya existan más algunos nuevos que se generen.

Este método se describe con mayor profundidad en el apartado Estructura de metapaquetes más abajo.

### 1.2.1.- ESTRUCTURA DE METAPAQUETES

Se ha desarrollado un sistema de generación automática de metapaquetes de configuración llamado CDD. Para entender la arquitectura de CDD es necesario aclarar previamente algunos conceptos básicos como los *metapaquetes*, *metapaquetes de configuración* y *componentes*.

Un *metapaquete* como tal es un paquete Debian vacío (no instala ningún fichero en el sistema salvo los mínimos imprescindibles que exige el sistema de paquetes de Debian) que depende, a su vez, de un conjunto de paquetes. De esta forma, la instalación del metapaquete provoca la instalación de todo el conjunto de paquetes del que depende.

Un *metapaquete de configuración* es un paquete cuya instalación tiene como objetivo configurar un

conjunto de aplicaciones a partir de la instalación de una serie de ficheros de configuración en el sistema. Estos ficheros de configuración dependen, lógicamente, de las aplicaciones concretas que se configuren y son totalmente dependientes de ellas.

Un *componente* es una combinación de metapaquetes y metapaquetes de configuración que se encargan de instalar y configurar adecuadamente una parte del sistema. Por ejemplo, se podría hablar de un componente "escritorio" que se encargue de instalar el sistema X-Window, Gnome, ofimática, etc y de configurarlo como interese.

Un componente está formado, a su vez, por otros subcomponentes que configuran partes más específicas dentro del área que abarca un componente. Por ejemplo, en un componente "escritorio" se podrían tener varios subcomponentes: oficina, multimedia, internet,...

Se puede ver todo esto de forma gráfica en la figura 4.

Observando la figura, se puede ver, en primer lugar, la existencia del metapaquete de Guadalinex v3. Éste es el metapaquete padre de la distribución y el objetivo es que lleve a cabo la instalación de todos los componentes que se han definido para Guadalinex v3. Por tanto, existe una dependencia de dicho paquete de todos los "hijos" de él (dependencias a nivel de paquete Debian).

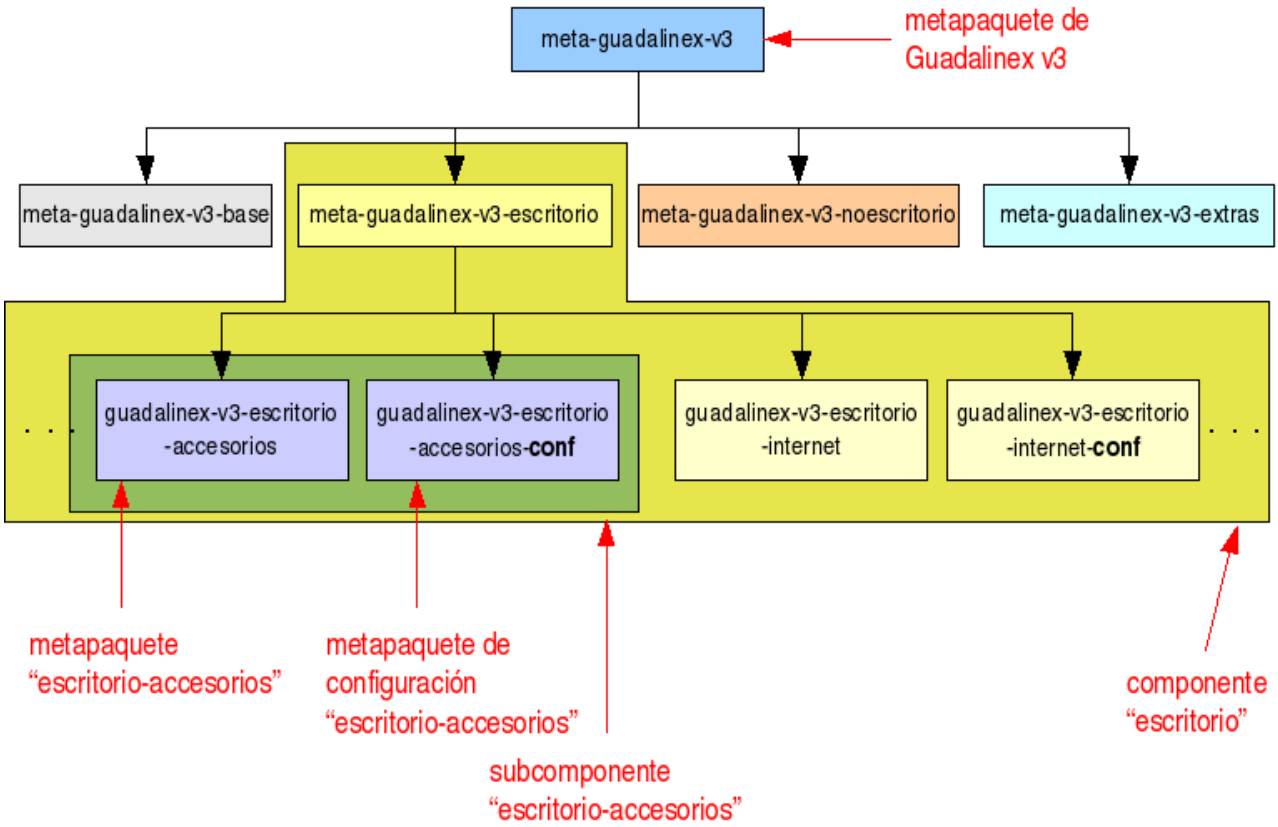



*Figura 4: Jerarquía de paquetes y metapaquetes de Guadalinex v3*

Por otra parte, se observa la existencia de cuatro componentes, a saber: "base", "escritorio", "noescritorio" y "extras". Cada uno de estos componentes está formado por un conjunto de subcomponentes. Se puede observar, por ejemplo, que el componente "escritorio" contiene, a su vez, los subcomponentes "accesorios" e "internet", además de otros que no se han detallado explícitamente en la figura.

Cada uno de los subcomponentes está formados por dos paquetes claramente diferenciados:

– Metapaquete, que lleva a cabo la instalación de todos los paquetes (aplicaciones) que forman parte del subcomponente. En el caso del metapaquete "escritorio-accesorios", incluye en sus dependencias paquetes tales como "file-roller", "gthumb", "tomboy",...

– Metapaquete de configuración, que lleva a cabo la configuración de las aplicaciones anteriores. En el caso del metapaquete de configuración "escritorio-accesorios-conf" se incluyen ficheros tales como "/usr/share/applications/file-roller.desktop", que configura la entrada del menú del paquete "file-roller".

CDD permite definir y generar de forma sencilla estructuras de metapaquetes y componentes similares a la anterior, pudiéndose adaptar a las necesidades particulares de la distribución final. Es posible definir el número de componentes y subcomponentes que se desee y cada uno de ellos puede instalar y configurar cuantas aplicaciones se quiera. Sin embargo, presenta un par de limitaciones:

– Una misma aplicación no puede estar en dos subcomponentes diferentes, porque esto provocaría problemas de dependencias entre ellos.

– El número de niveles en los que se ha estructurado la jerarquía anterior no se puede modificar; por tanto, no se puede ampliar o disminuir este número. Lo que sí se puede hacer, como ya se ha comentado, es añadir o quitar cuantos componentes se desee.

## 1.2.2.- DESCRIPCIÓN FUNCIONAL DE "CDD"

CDD es una utilidad para crear paquetes de configuración y metapaquetes de una forma sencilla. El diseño de CDD se apoya en dos pilares fundamentales:

– Sistema de paquetes de Debian: este sistema proporciona realmente la base sobre la que se sustenta CDD. Permite establecer relaciones de dependencia de los metapaquetes entre sí y con los paquetes hijo que instalan, llevar a cabo un control de versiones, instalar aplicaciones, ejecutar comandos antes o después de la instalación de paquetes, y un largo etcétera.

– "dpsyco": El sistema dpsyco se utiliza como base para generar los metapaquetes de configuración. Hay que tener en cuenta que el sistema de paquetes de Debian impide que un paquete sobreescribe la configuración de otro, por lo que no generar metapaquetes de configuración usando el sistema de paquetes de Debian tal cual.

Mediante “dpsyco”, se pueden generar metapaquetes de configuración que sí llevan a cabo la configuración de otros paquetes instalados en el sistema.

Para ello, “dpsyco” funciona de la siguiente manera: dentro de un metapaquete de configuración, se almacena en un “skel” toda la configuración de aplicaciones que se vaya a llevar a cabo. Cuando se instala dicho metapaquete, todos estos ficheros del “skel” se almacenan en “/usr/share/dpsyco”. Cuando se lleva a cabo la “postconfiguración” del metapaquete se produce la ejecución de “update-dpsyco-skel” (una de las herramientas que proporciona “dpsyco”) de manera que se buscan los ficheros originales de configuración del sistema, se guarda una copia de seguridad de los mismos (en “/var/lib/dpsyco-skel”) y después se sustituyen por los nuevos del metapaquete. De esta forma, las aplicaciones quedan configuradas como se desea pero almacenando previamente una copia de seguridad de la configuración anterior. Si se desinstala el metapaquete de configuración, se restaura la copia anterior de los ficheros de configuración.

Todo esto se puede ver de forma gráfica en la figura 5.

CDD se ha diseñado para generar un conjunto de paquetes con ciertas relaciones de dependencia entre ellos. CDD trata cada componente de forma independiente de manera que al final se genera un metapaquete por encima que instala todos los componentes que formen parte de la distribución final.

La relación de dependencias en el caso de un componente concreto se muestra en la figura 6.

Como se observa, el metapaquete padre del componente tiene dos tipos de dependencias con los hijos. El primero de estos tipos es una “predependencia” con los metapaquetes de los subcomponentes hijos. El segundo tipo es una dependencia normal de los metapaquetes de configuraciones de cada subcomponente. De esta manera se asegura que al instalar el componente, primero se instalan todas las aplicaciones que forman parte del mismo y después se produce la configuración.

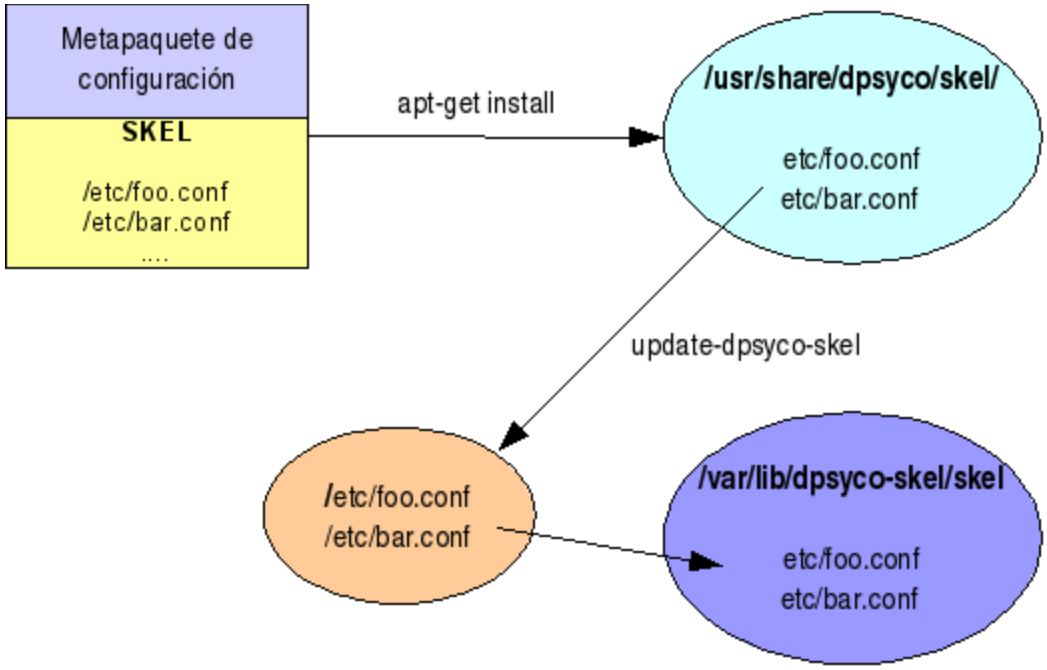



*Figura 5: Funcionamiento de "dpsyco" durante la instalación de un metapaquete de configuración.*

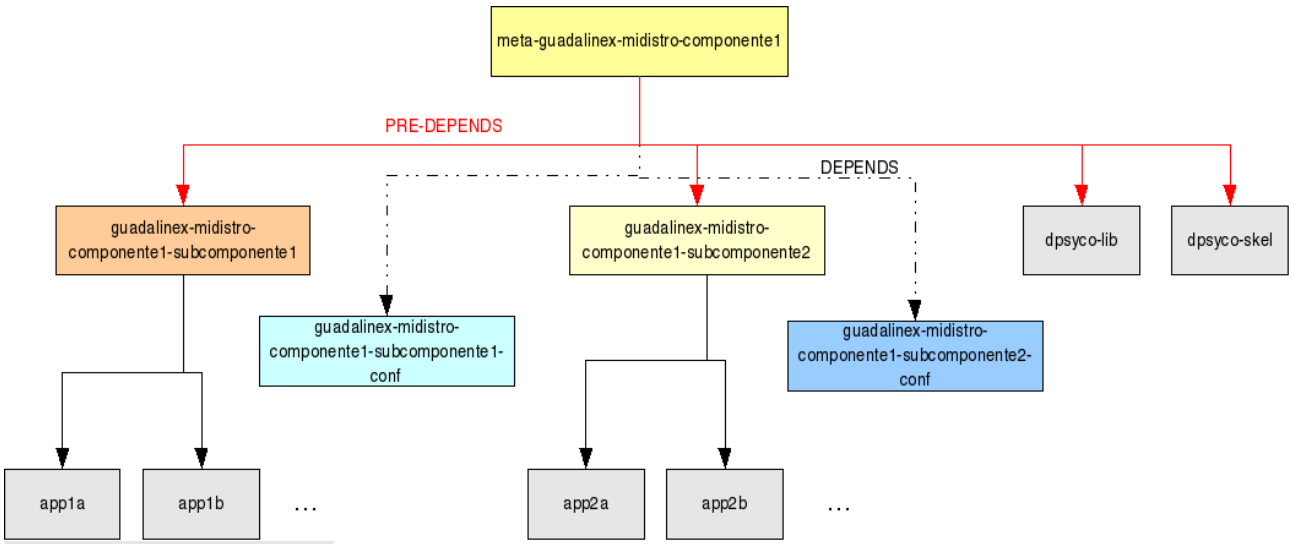



*Figura 6: Estructura de un componente generado con CDD.*

## 1.2.3.- LA APLICACIÓN "CDD".

CDD se encuentra en el directorio "apps_sel_and_conf/cdd" del repositorio de Guadalinex. Esto es lo que contienen los diferentes directorios:

- "build/gen-package": Contiene scripts necesarios para generar los paquetes. Además, existe un directorio "debian" con las plantillas de todos los ficheros necesarios para generar los paquetes Debian. Estas plantillas se utilizarán por defecto en la generación de los paquetes. Aunque es posible crear ficheros diferentes para algún componente concreto. En este caso, el fichero que se cree sustituirá al que se incluye por defecto. Esto se verá más adelante.

- "build/packages/": Contiene los paquetes generados.

- "conf" -> En este directorio es donde se configuran los distintos paquetes. Contiene un directorio para cada componente, que, a su vez, contiene un directorio para cada subcomponente. Un ejemplo sería:

```
conf/
|-- v3-escritorio (componente "escritorio" de Guadalinex v3)
|   |-- accesorios
|   `-- ...
|
`-- v3-mini-base (componente "base" de Guadalinex v3 mini)
    `-- base
          |-- DEBDEPENDS
          |-- debian
          `-- skel
                `-- ...
```

Si nos centramos en el componente “v3-mini-base”, se ve que éste contiene un único subcomponente “base”. Se va a describir la estructura de directorios para el caso de uno de los subcomponentes, en este caso, “base”:

- “conf/v3-mini-base/base/DEBDEPENDS”: Contiene una lista de paquetes de los cuales depende el subcomponente "base", separados por comas, de manera que al instalarse este paquete se instalarán todas las dependencias.

- “conf/v3-mini-base/base/debian”: En este directorios se pueden opcionalmente colocar algunos de los típicos scripts del sistema de paquetes de Debian: “preinst”, “postinst”, “prerm” y “postrm”. En caso de que se incluya alguno de estos scripts, se sustituirá el que se utiliza por defecto en la plantilla, sólo en el metapaquete de configuración (no en el metapaquete de dependencias). El más utilizado de estos scripts suele ser “postinst” para realizar algunas tareas de configuración finales a partir de la ejecución de algunos comandos.

- “conf/v3-mini-base/base/skel “: Contiene los ficheros de configuración que se copiarán al sistema, manteniendo la estructura de directorios. Por ejemplo, el fichero “conf/mini/base/skel/etc/issue” se instalará en el sistema en “/etc/issue”.

## 1.2.4.- PROCESO DE GENERACIÓN DE LOS PAQUETES.

Antes que nada, es **muy importante** tener en cuenta en todo momento que antes de generar algún paquete es necesario que *todos* los ficheros estén subidos en el svn.

Veamos paso a paso cómo construir un paquete desde cero. Sirva de ejemplo la construcción de una nueva distribución ficticia “midistro” (que sería una variante de la distribución padre, como podría ser por ejemplo, “mini”,la variante para equipos antiguos de Guadalinex). El metapaquete padre de “midistro” depende de dos paquetes, uno al que llamaremos “componente1” y otro que bautizaremos como “componente2”. Éstos serían los dos componentes que forman parte de la distribución ficticia “midistro”.

El primer componente está formado, a su vez, por dos subcomponentes, “subcomponente1” y “subcomponente2”. El primero de ellos, instalará las aplicaciones “app11a” y “app11b”. El segundo, las aplicaciones “app21a” y “app21b”. Cada una de estas aplicaciones se configurará con un fichero de configuración que llevará por nombre “/etc/app???.conf”, correspondiente cada uno a cada aplicación. Esta misma estructura se repetirá para el “componente2”.

En este escenario, el procedimiento para la generación de los paquetes sería:

1) Creamos la estructura de directorios con la configuración de cada componente, tal y como se muestra a continuación (Es importante recordar que todos los elementos que forman parte del árbol anterior deben estar subidos al repositorio de subversion).

```
conf/midistro-componente1
|-- subcomponente1
|    |-- DEBDEPENDS
|    |-- debian
|    `-- skel
|        `-- etc
|            |-- app11a.conf
|            `-- app11b.conf
`-- subcomponente2
    |-- DEBDEPENDS
    |-- debian
    `-- skel
        `-- etc
            |-- app12a.conf
            `-- app12b.conf
```

2) Guardamos en los ficheros DEBDEPENDS de cada componentes las listas con las dependencias de paquetes que debe instalar cada uno de los componentes. Por ejemplo, en "subcomponente1/DEBDEPENDS" se tendrá: "app11a, app11b".

3) Creamos los ficheros "packages-midistro-componente1" y "packages-midistro-componente2" en "build/gen-package", con el siguiente contenido:

   subcomponente1

   subcomponente2

   Se está indicando que cada componente de la distribución contiene dos subcomponentes, "subcomponente1" y "subcomponente2".

4) Modificamos el fichero "build/gen-package/Makefile" para que contenga lo necesario para crear el nuevo paquete:

   Contenido antes de modificar (suponemos que ya existe un paquete llamado v3):

```
v3: packages-v3-base packages-v3-escritorio packages-v3-noescritorio
packages-v3-extras
        ./build ../../conf/v3-base
        ./build ../../conf/v3-escritorio
        ./build ../../conf/v3-noescritorio
        ./build ../../conf/v3-extras
        ./buildmeta meta-guadalinex-v3-base meta-guadalinex-v3-
escritorio meta-guadalinex-v3-noescritorio meta-guadalinex-v3-extras

all: v3
```

Después:

```
v3: packages-v3-base packages-v3-escritorio packages-v3-noescritorio
packages-v3-extras
      ./build ../../conf/v3-base
      ./build ../../conf/v3-escritorio
      ./build ../../conf/v3-noescritorio
      ./build ../../conf/v3-extras
      ./buildmeta     meta-guadalinex-v3-base     meta-guadalinex-v3-
escritorio meta-guadalinex-v3-noescritorio meta-guadalinex-v3-extras

midistro: packages-midistro-componente1 packages-midistro-componente2
        ./build ../../conf/midistro-componente1
        ./build ../../conf/midistro-componente2

all: v3 midistro
```

Así se está indicando que para construir “midistro” hay que construir los componentes que forman parte ella.

5) Desde el directorio “build/gen-package” ejecutamos:

“make” (para construir todas los paquetes de todas las distribuciones definidas), o bien

“make midistro” (para construir los paquetes sólo de la distribución “midistro”)

6) En el directorio “build/packages/binary-i386/” se encuentran los paquetes ya creados. En este caso concreto, se tendrían los siguientes paquetes generados:

```
binary-i386/midistro-componente1
|-- guadalinex-midistro-componente1-subcomponente1-conf_1.0-r1955_all.deb
|-- guadalinex-midistro-componente1-subcomponente1_1.0-r1955_all.deb
|-- guadalinex-midistro-componente1-subcomponente2-conf_1.0-r1955_all.deb
|-- guadalinex-midistro-componente1-subcomponente2_1.0-r1955_all.deb
`-- meta-guadalinex-midistro-componente1_1.0-r1955_all.deb
binary-i386/midistro-componente2
|-- guadalinex-midistro-componente2-subcomponente1-conf_1.0-r1955_all.deb
|-- guadalinex-midistro-componente2-subcomponente1_1.0-r1955_all.deb
|-- guadalinex-midistro-componente2-subcomponente2-conf_1.0-r1955_all.deb
|-- guadalinex-midistro-componente2-subcomponente2_1.0-r1955_all.deb
`-- meta-guadalinex-midistro-componente2_1.0-r1955_all.deb
```

7) El último paso consiste en la generación del metapaquete padre de la distribución, que provocará la instalación de los demás. El primer paso consiste en crear el script “build/gen-package/buildmeta-midistro” a partir de, por ejemplo, “build/gen-package/buildmeta”. Hay que editar el fichero una vez creado y cambiar el contenido para adaptarlo al paquete de la nueva distribución. En particular, hay que cambiar:

   1. DEBIANSRC=”meta-pkg-midistro”

   2. Todas las ocurrencias de “meta-guadalinex-v3” y las referencias a ella por las correspondientes a la nueva distribución, de manera que quedaría como “meta-guadalinex-midistro”.

   Al final, el contenido de este fichero quedaría como sigue:

```
#!/bin/bash

. cdd.cfg
DEBIANSRC="meta-pkg-midistro"

if [ ! $# -gt 0 ];then
        echo "Usage: $0 depend [depend..]"
        exit 1
fi

cd $DEBIANSRC

echo "Source: meta-guadalinex-midistro
Section: misc
Priority: optional
Maintainer: Junta de Andalucía <packmaster@guadalinex.org>
Standards-Version: 3.6.0

Package: meta-guadalinex-midistro
Architecture: all" > debian/control

depends=$(echo $@ | sed -e 's/ /, /g')
echo "Depends: $depends" >> debian/control

echo "Description: Guadalinex midistro metapackage
 This metapackage installs a whole Guadalinex midistro distribution." >> debian/control

last_revision=$(cat ../../../conf/.svn/entries |grep revision|sed -e 's/^.*revision="\([0-
9]*\)"\/>/\1/')
dch -v 1.0-r${last_revision} "Auto-generated version increment" || continue

fakeroot debian/rules binary
fakeroot debian/rules clean
cd ..
mv meta-guadalinex-midistro*.deb $CDDPACKAGESDIR
```

El segundo paso cosiste en crear el directorio “meta-pkg-midistro” con el directorio “debian” dentro. En él, hay que crear los ficheros “changelog” y “rules” a partir de los que hay en “meta-pkg/debian”. En “changelog” debe haber una única entrada correspondiente a la versión inicial del paquete junto con el nombre. El fichero “rules” no hay que modificarlo.

Por último, hay que modificar el fichero “gen-packages/Makefile” para añadir una línea para generar el metapaquete padre. La parte correspondiente a “midistro” quedaría así:

```
midistro: packages-midistro-componente1 packages-midistro-componente2
        ./build ../../conf/midistro-componente1
        ./build ../../conf/midistro-componente2
           ./buildmeta-midistro meta-guadalinex-midistro-componente1 meta-
guadalinex-midistro-componente2
```

Ya sólo queda volver a ejecutar el comando para la generación, es decir, “make midistro”, y se obtendrán los paquetes en “build/packages/binary-i386”.

Es importante señalar que en el fichero “build/gen-package/cdd.cfg” se pueden personalizar diversos elementos del sistema de paquetes, como por ejemplo:

- CDDFLAVOUR: Nombre del proyecto (por ejemplo: “guadalinex”)
- CDDMAINTAINER: Mantenedor de los paquetes.
- CDDSVN: URL del repositorio de subversion del proyecto.

## 1.2.5.- GENERACIÓN DE SUPLEMENTOS.

Los suplementos constituyen un mecanismo para añadir software adicional a Guadalinex v3 de forma sencilla para el usuario final. Un suplemento está formado por un conjunto de paquetes. Al insertar el CD en el sistema, éste se reconocerá automáticamente, se notificará al usuario y habilitará un mecanismo sencillo para su instalación.

Para facilitar la tarea de generación de suplementos a aquéllos que estén interesados, se ha desarrollado una aplicación que lleva por nombre “Suppletory Disk Generator (SDG)” y que permite llevar a cabo esta tarea de forma sencilla. Esta aplicación se encuentra en “other_apps/suppletory_disk_generator”.

El primer paso para llevar a cabo la generación de un suplemento consiste en generar el paquete Debian para instalarlo en el sistema. Esto se hace ejecutando, desde el directorio raíz de la aplicación en el repositorio, el comando:

```
$ fakeroot dpkg-buildpackage
```

El resultado será la generación del paquete “../guadalinex-suppletory-disk-generator_0.1-1_i386.deb”.

El siguiente paso será instalar el paquete en el sistema:

```
$ sudo dpkg –install guadalinex-suppletory-disk-generator_0.1-1_i386.deb
```

A partir de ahí, ya tenemos el sistema preparado para poder generar suplementos.

## PROCEDIMIENTO DE GENERACIÓN.

1) Crear un directorio donde se va a construir el suplemento, por ejemplo:

```
$ mkdir ~/mi_suplemento
```

2) Creación del esqueleto del suplemento:

```
$ sdg_new
```

Se generará un árbol de directorios con la estructura siguiente:

```
.
|-- applications.menu
|-- config
|-- debs
|-- icon.png
`-- master
    `-- guadalinex-suplementos-apps
        |-- gnome-app-install.glade
        |-- icons
        `-- template.html
```

3) Configuración del suplemento:

1. El fichero “config” se utiliza para configurar el suplemento, a través de las siguientes variables:

    1. NAME xxxx: Aquí se configura el nombre del suplemento.

    2. SHORTNAME xxxx: Se utiliza para concatenarse al nombre de la ISO generada. No debe tener más de ocho caracteres.

    3. DESC xxxx: Descripción del suplemento (una línea).

    4. METAPACKAGE nombre-del-metapaquete: Para la instalación de un suplemento hay

que generar un metapaquete que dependa de todos los paquetes que deseen instalarse a partir del suplemento.

5. PACKAGES: Lista de paquetes, separados por espacios, que estarán disponibles para instalarse de forma independiente además del metapaquete del suplemento. Se puede utilizar para crear sub-suplementos o incluso llegar a la posibilidad de instalación de paquetes independientes.

6. URI repositorio: Mediante esta variable se pueden especificar repositorios externos de paquetes para descargar aquellas dependencias de los paquetes del suplemento que no se encuentren en el propio CD-ROM. Al menos debe existir una línea URI (aunque sea vacía). Aunque se pueden especificar tantos repositorios como se desee.

   El formato es el estándar de los URIs de Debian:

   URI http://repositorio.guadalinex.org/guadalinex-flamenco main universe multiverse

2. Directorio “debs”: En este directorio se deben colocar todos los paquetes que forman parte del suplemento (incluyendo el metapaquete padre). Si algún paquete tiene una dependencia que no se incluye en el CD-ROM, se intentará descargar de los repositorios especificados en las “URIS” durante la instalación del suplemento.

3. Fichero “icon.png”: Icono que se mostrará en el sistema cuando se introduzca el CD-ROM del suplemento generado. Hay que cuidar el tamaño de la imagen para evitar tamaños demasiado grandes (tamaño recomendado: 52x52 píxeles).

4) Generación del suplemento: Una vez se ha configurado el suplemento como se desea, el último paso consiste en generar la imagen ISO para grabar el CD-ROM. Esto se hace a través de la ejecución del comando:

```
$ sdg_build
```

Este comando hay que ejecutarlo desde el directorio raíz del suplemento. Tras la ejecución, se obtendrá un fichero .ISO con la imagen del suplemento en el directorio raíz del suplemento.

# 2.- ASPECTOS MODIFICABLES DE LA DISTRIBUCIÓN.

## 2.1.- NOMBRE DE LA DISTRIBUCIÓN.

A lo largo de la distribución podemos encontrar en varios lugares el nombre de Guadalinex. A continuación se describe los lugares con mayor relevancia que pueden ser cambiados:

### 2.1.1.- /ETC/MOTD

motd son siglas de Message Of The Day, el contenido de este fichero es mostrado tras hacer login de forma correcta en el sistema operativo Guadalinex. En Guadalinex, este fichero se modifica desde el paquete "guadalinex-v3-base-base-conf", en concreto, el fichero en el repositorio se encuentra en "guadalinex2005/trunk/apps_sel_and_conf/cdd/conf/v3-base/base/skel/etc/motd".

### 2.1.2.- /ETC/ISSUE

Este fichero contiene el texto, normalmente usado como identificador del sistema operativo, que será mostrado junto al login. Se encuentra en el mismo paquete que "/etc/motd", puede encontrarse en "guadalinex2005/trunk/apps_sel_and_conf/cdd/conf/v3-base/base/skel/etc/issue".

### 2.1.3.- /ETC/LSB-RELEASE

Al igual que los archivos de apartados anteriores, comparten el mismo directorio, por tanto se encuentra en "guadalinex2005/trunk/apps_sel_and_conf/cdd/conf/v3-base/base/skel/etc/lsb-release". Dicho fichero posee información sobre la distribución, en concreto su contenido es el siguiente:

```
DISTRIB_ID=Guadalinex

DISTRIB_RELEASE=v3.0.1

DISTRIB_CODENAME=flamenco

DISTRIB_DESCRIPTION="Guadalinex v3"
```

Es importante destacar que si se quisiera cambiar el contenido de la variable DISTRIB_ID implicaría tocar un fichero perteneciente a gnome-system-tools en el cual lista las distribuciones de confianza. Sin alterar dicho fichero, el escritorio estaría "capado" para realizar ciertas funciones puesto que gnome-system-tools no se relaciona bien con las distribuciones no listadas en su base de datos de confianza. No sólo gnome-system-tools depende de esta información, por tanto habría que modificar estos datos con sumo cuidado.

Para ello, habría que tocar el fichero "/usr/share/setup-tool-backends/scripts/platform.pl" que puede encontrarse en los metapaquetes de guadalinex, concretamente en "guadalinex2005/trunk/apps_sel_and_conf/cdd/conf/v3-escritorio/gnome/usr/share/setup-tool-backends/scripts/platform.pl" y añadirle el nombre deseado.

Breve extracto del actual platform.pl:

```
...
"freebsd-6"       => "FreeBSD 6",
"gentoo"          => "Gentoo Linux",
"guadalinex-2005" => "Guadalinex",
"vlos-1.2"        => "Vida Linux OS 1.2",
"archlinux-0.7"   => "Arch Linux 0.7",
...
```

## 2.2.- IMAGEN.

Hay diferentes partes en una distribución que se pueden cambiar y personalizar, pero los elementos que siempre querremos tocar serán aquellos relacionados con la *imagen* de la distro. Es decir, todos aquellos elementos gráficos que le dan una identidad a la distribución y la diferencian visualmente de otra. Esto es lo que suele llamarse *artwork*.

Dicho *artwork* puede tener que ver con la distribución una vez instalada, como con elementos gráficos del CD (tanto *live* como *no live*). Aquí mostraremos los más importantes y de forma genérica, pero en otros apartados profundizaremos en alguno de ellos.

### 2.2.1.- ISOLINUX.

*Syslinux* es un gestor de arranque pensado para dispositivos extraíbles, preferentemente. Pero Syslinux contiene programas y configuraciones específicos para algunos de estos dispositivos. Por ejemplo, para los CSS, tiene el *Isolinux*, que es el que nosotros usaremos porque está especialmente diseñado para CDs.

No entraremos en muchos detalles en la forma de trabajar de *Syslinux* o *Isolinux*, pero si explicaremos lo suficiente para entender sus opciones de configuración y poder modificarlo, tanto para el sistema *Live* como para el *No Live*.

El archivo de configuración se llama *isolinux.cfg* y tiene una pinta parecida a esta:

```
default linux
DISPLAY isolinux.txt

TIMEOUT 100
PROMPT 1

F1 isolinux.txt
F2 ayudaf2.txt
F3 ayudaf3.txt
F4 ayudaf4.txt
F5 ayudaf5.txt
F6 ayudaf6.txt
F7 ayudaf7.txt
```

```
F8 ayudaf8.txt
F9 ayudaf9.txt

KBDMAP es.kbd
FONT fuente.psf

label linux
  kernel vmlinuz
   append ramdisk_size=100000 root=/dev/ram0 initrd=initramfs BOOT=live
debug splash quiet
label db
  kernel vmlinuz
   append ramdisk_size=100000 root=/dev/ram0 initrd=initramfs BOOT=live
debug break
```

Éste es un archivo de configuración de *isolinux*, que como explicamos más anteriormente, es el encargado de hacer que el ordenador sepa que debe arrancar desde el CD y donde está todo lo necesario (sector de arranque, kernel, initramfs) para que pueda hacerlo.

La configuración del *isolinux* no es muy diferente de la de otros gestores de arranque como *LILO* o *Grub*. Veamos primero las partes comunes a estos otros sistemas:

- **default**: Es la opción predeterminada para arrancar, a menos que se especifique otra cosa en tiempo de arranque.
- **timeout**: Es el tiempo que esperará antes de arrancar. Esto brinda de unos segundos (en caso de que esté especificado) al usuario para seleccionar otro arranque u otras opciones. Este contador se desactivará en cuanto el usuario pulse cualquier tecla.
- **prompt**: Indica si se muestra al usuario una posible entrada de parámetros (*boot:* ) o no. *1* indica que se muestre y *0* que no se muestre, a no ser que se pulse las teclas *Alt* o *Mayúsculas*
- **label**: Sirve para indicarle un nombre a la opción. De esta manera se puede llamar directamente desde la línea de comandos mostrada (*boot:*).
- **kernel**: Indica cuál es el nombre y *path* del kernel.

- **append**: Con esta opción se le indican los parámetros que se le pasarán al kernel en el arranque. Éstos quedarán guardados luego en */proc/cmdline* para ser usados por el propio kernel u otros programas.

  Ahora algunos parámetros especiales del *syslinux/isolinux*:

- **F#** (F1, F2, F3...): Con estas marcas especiales, indicamos que archivo mostrar en caso de que el usuario pulse dichas teclas (F1, F2, F3...). Si no definimos alguna tecla, no se mostrará nada al ser ésta pulsada. Generalmente se usan para mostrar ayudas.
- **kdbmap**: Le indica la localización y nombre de el mapa de teclado que se desea usar. Esto es útil para poder usar caracteres especiales y para que coincidan las teclas con nuestros teclados, en vez de los teclados en inglés.
- **font**: Con esta opción podemos indicarle que se use unas determinados tipos de letras, en vez de las que se usan por defecto.

Un detalle importante es que no diferencia entre mayúsculas y minúsculas, así que es lo mismo poner *TIMEOUT* o *timeout*.

Ahora que conocemos qué es el *Isolinux* y como es su archivo de configuración, vamos a ver qué podemos personalizar y cómo.

Tenemos varios elementos personalizables en el *Isolinux*:

- **Opciones del isolinux.cfg**: Esta parte es sencilla, simplemente tenemos que ver que elementos de los que se definen nos interesa dejar como están y cuáles queremos personalizar. Cosas típicas son la ayuda (los archivos que se conectan a *F#*) y el de la pantalla principal (*isolinux.txt*), en donde aparece el nombre, versión y algún detalle más de la distribución.
- **Imagen del arranque** (*/media/distro/master/isolinux/splash.rle*): La imagen es una de las principales cosas que querremos cambiar, pero éste no es una imagen normal. Tiene unas características concretas que debemos cumplir. Vamos a ver las características de este tipo de imágenes y después el proceso a seguir para conseguirla.
  - **Características**: Imagen en formato *lss16* de resolución *640 x 400* en *modo indexado* a *16 colores*.
- **Proceso**:

**1-** Crear una imagen en el *Gimp* (se puede hacer en otros, pero nos será más fácil con éste) de tamaño *640 x 400*.

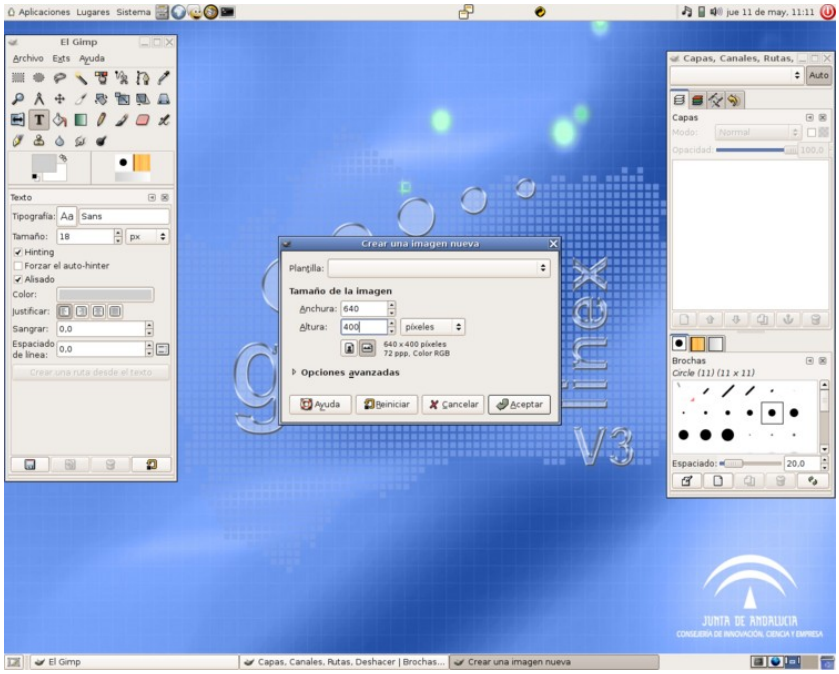



*Figura 7: Crear imagen nueva*

## **2-** Editar la imagen

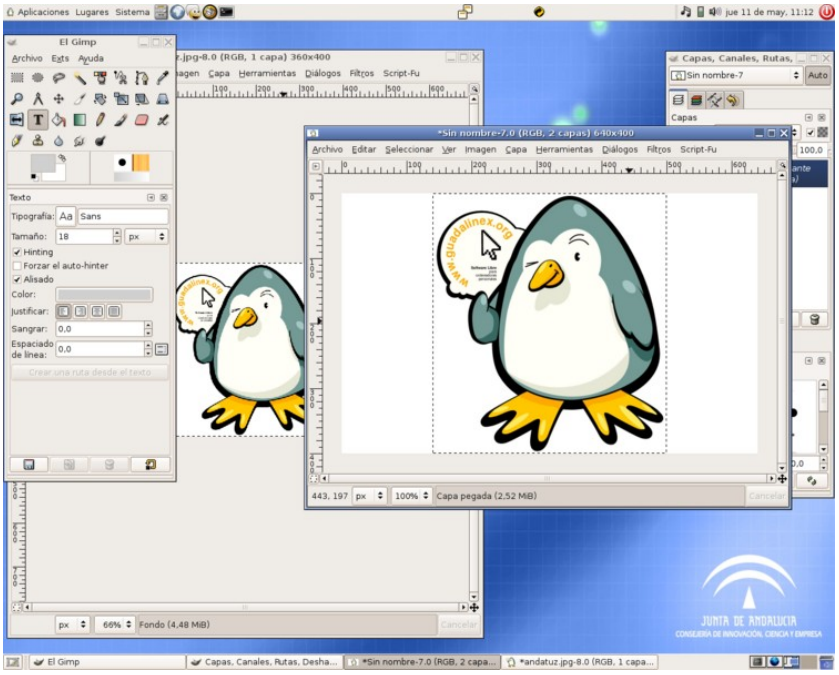
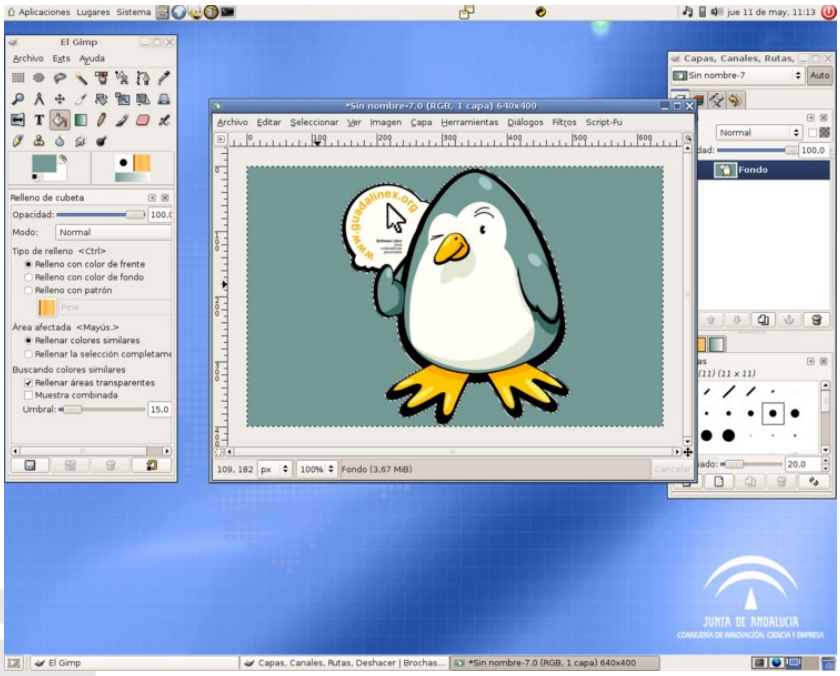
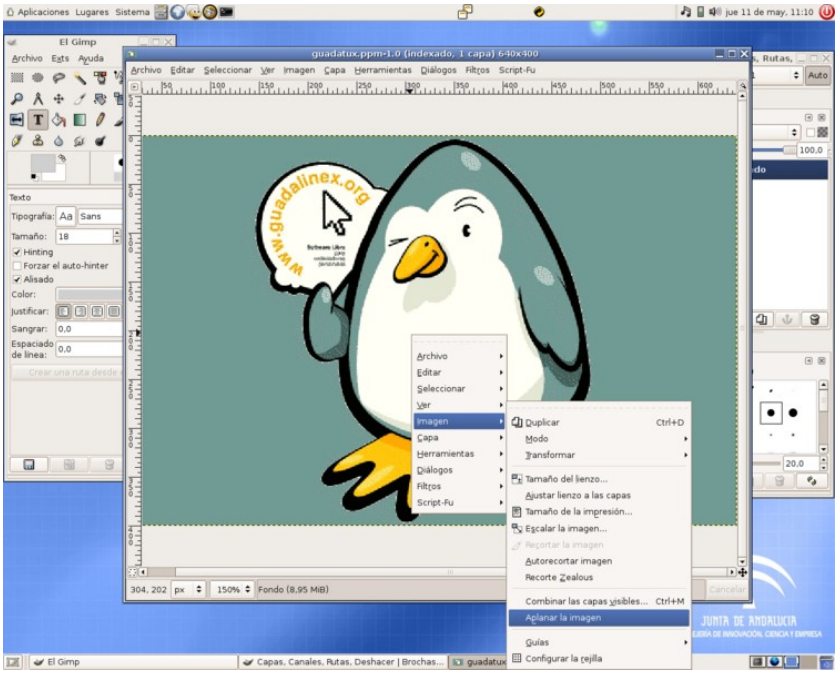


*Figura 8: Copiando logotipo en la imagen recién creada desde otro archivo.*

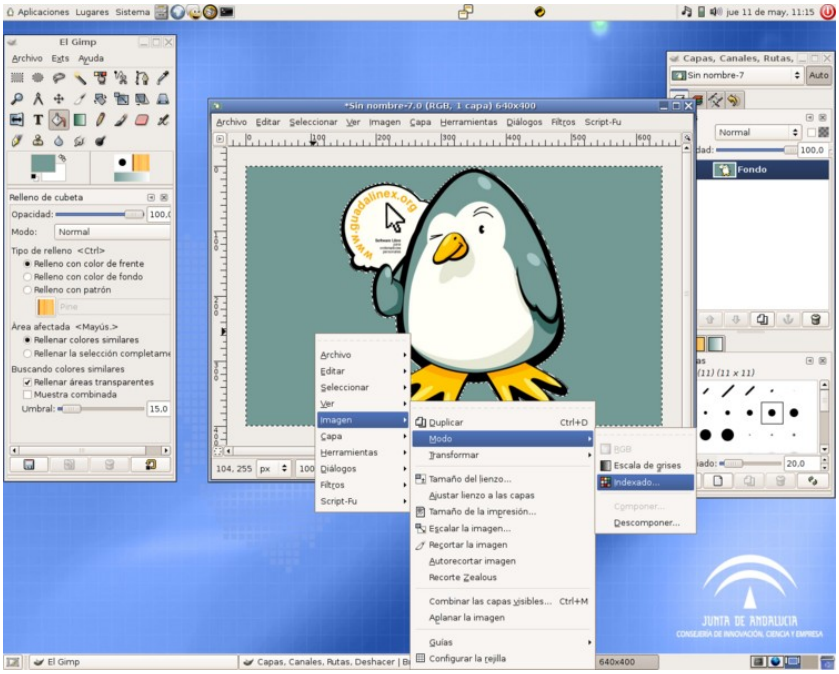
*(Esto es un ejemplo)*

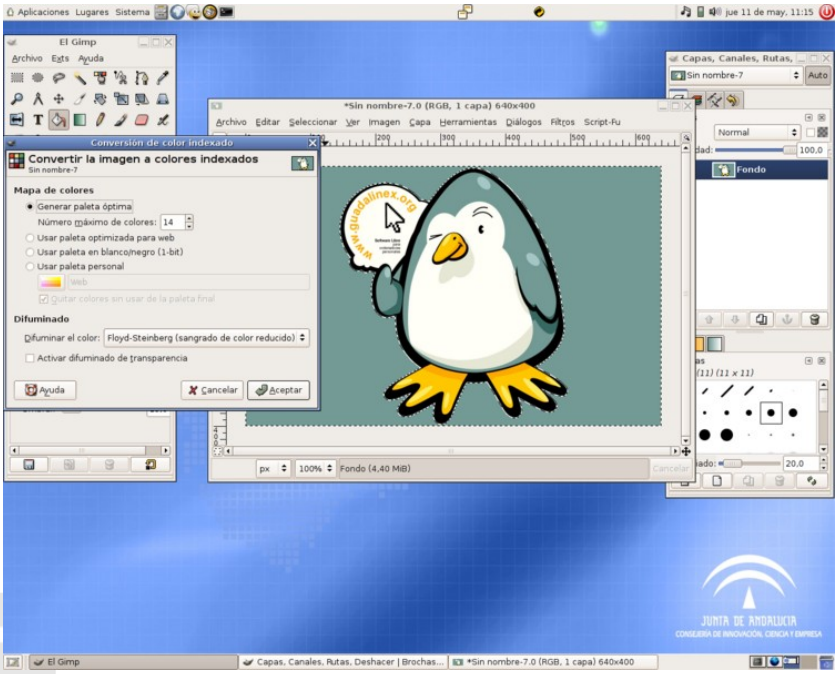


**3-** Aplanar la imagen.

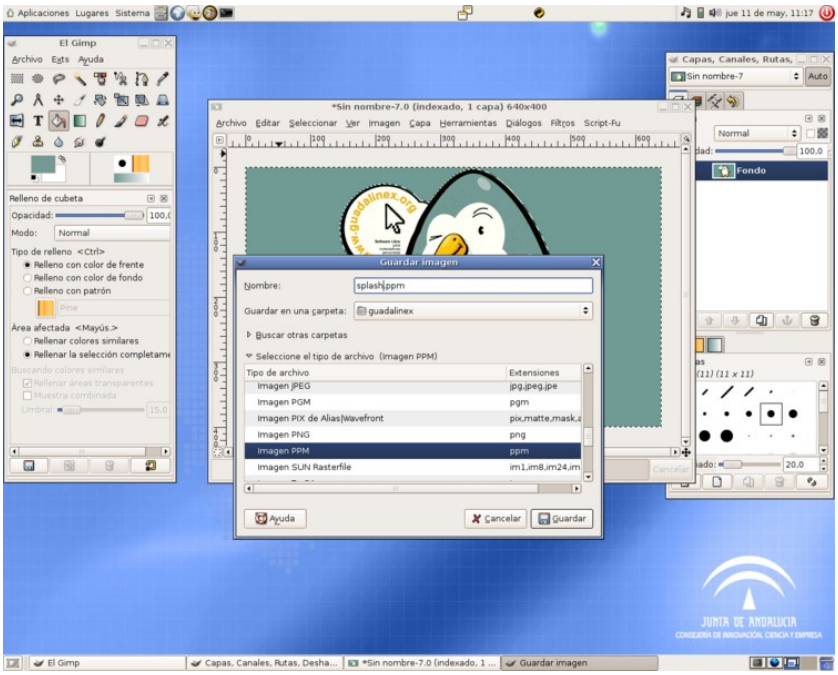


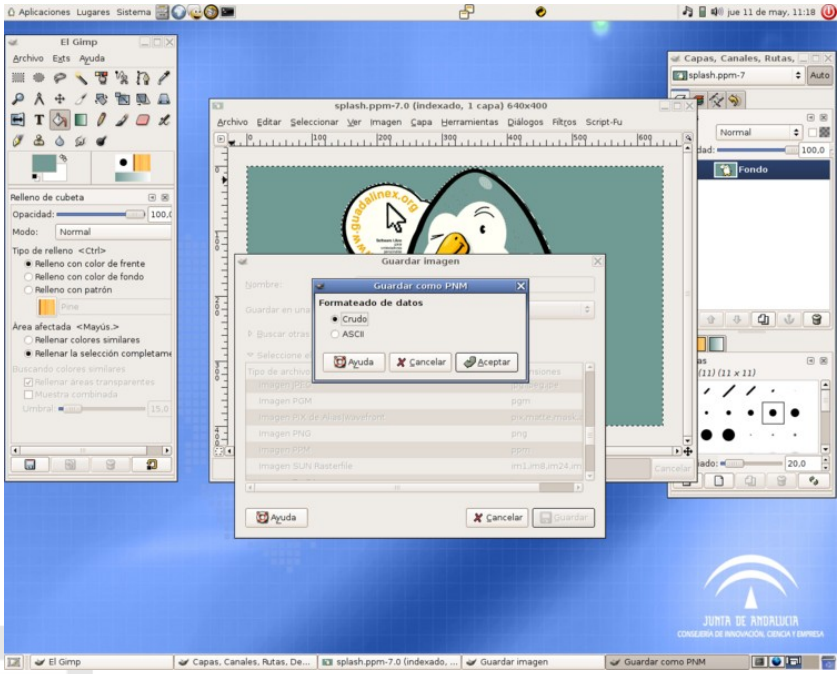
**4-** Convertir a *modo indexado* con **14** colores. Dejaremos 2 para el texto y el isolinux. Así evitaremos sorpresas.



**5-** Guardar como archivo *PPM*



**6-** Convertir en formato *.rle*. Con el programa *ppmtolss16*, contenido en el paquete *syslinux*, pasaremos nuestro archivo de un formato a otro:

```
$ ppmtolss16 '#d0d0d0=7' < splash.ppm > splash.rle
```

Con el **#d0d0d0=7** estamos definiendo que el color número **7** de la *paleta de colores*, que es el que usa el *isolinux* para el texto, se fije al color, definido en hexadecimal, *#d0d0d0*.

**7-** Ya tenemos nuestro archivo *splash.rle*, sólo nos queda guardarlo en los directorios adecuados tanto para el CD *live*, como el *no live*.

Para hacer pruebas rápidas sobre los cambios hechos en la imagen o en los archivos de ayuda del isolinux, podemos generar una iso pequeña y probarla con el software de emulación *qemu* (instalable fácilmente con: *sudo apt-get install qemu*). Con el siguiente script se pueden hacer pruebas muy rápida y fácilmente:

```
#!/bin/sh

LIVE_DIR="/usr/share/genlive/isolinux/"
NO_LIVE_DIR="$CDIMAGE_ROOT/uda/isolinux/"

ISOLINUX_DIR="$LIVE_DIR"

mkdir /tmp/master
cp -a $ISOLINUX_DIR/{isolinux.*,f*,ayuda*} /tmp/master/
cp -a /usr/lib/syslinux/isolinux.bin /tmp/master/

mkisofs -l -r -J -V "image test" -hide-rr-moved -v -b isolinux.bin -c
boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o
/tmp/image_test.iso /tmp/master/

qemu -m 32 -cdrom /tmp/image_test.iso
```

Sólo hay que tener en cuenta la variable *ISOLINUX_DIR*, cuyo valor será *$LIVE_DIR* o *$NO_LIVE_DIR* en función de qué CD queramos probar. También habrá que asegurarse de sustituir los archivos originales por los que hemos modificado nosotros (*f#*, *isolinux.txt*, *ayudaf#.txt*, *splash.rle*, etc).

### 2.2.2.- USPLASH.

Una vez se ha seleccionado la "Guadalinex", en el "grub", para arrancar, lo siguiente que veremos y podremos personalizar será el "usplash".

El "usplash" es el programa que muestra el arranque en modo gráfico.

La imagen a usar tiene unas características similares a las del la imagen para el isolinux: 640x480 indexada a 16 colores.

### 2.2.3.- GRUB.

El "Grub" es el gestor de arranque usado por "Guadalinex" y lo primero que aparece (tras la BIOS) al arrancar un equipo instalado con ésta o una derivada.

Por esta razón es otro elemento visible que querremos personalizar.

La personalización hecha para "Guadalinex" consiste en cambiar el título de la entrada para el nuevo sistema instalado y aquellos que se encontraran. Aparte de ponerle una imagen de fondo.

Ambos cambios se pueden ver en el archivo de configuración del "grub": "/boot/grub/menu.lst".

Título:

```
title Guadalinex, kernel 2.6.12-9-386
```

Imagen:

En los capítulos dedicados al sistema "live" y al "no live" se explicará dónde tocar para instalar y mantener estos cambios.

# 2.3.- ESCRITORIO GNOME.

La mayor parte de las distribuciones, hoy en día, usan algún escritorio, para hacerle la vida mas fácil al usuario.

Por diversas razones, para Guadalinex, se eligió el escritorio Gnome. Por eso dedicaremos éste apartado a dicho escritorio.

Para profundizar más en este tema recomendamos mirar la documentación oficial[1] (en inglésś), ya que aquí sólo se darán unas pinceladas por los elementos más frecuentemente cambiados para una distribución como Guadalinex. Hay que tener en cuenta que esa guía es para la versión siguiente a la que está instalada en Guadalinex V3, por lo que puede que algunas cosas no estén todavía operativas en nuestra distro.

También puede ser de mucha utilidad información extra un artículo[2] sobre el sistema de configuraciones de Gnome, que complementará la guía anterior y la información aquí dada.

Veamos cómo personalizar las partes más relevantes.

## 2.3.1.- SELECTOR DE SESIÓN (GDM)

Aunque el "GDM" no es estrictamente parte del escritorio, si es parte de Gnome y es personalizable. Como casi todas las partes estéticas de Gnome, el GDM está basado en un sistema de temas. Lo único que necesitamos para cambiarle la apariencia al GDM, es crearle un tema y configurárselo.

No vamos a explicar como crear o modificar un tema para el GDM porque se sale del alcance de esta documentación.

El nuevo tema se guardaría en:

```
/usr/share/gdm/themes/[TEMA]
```

## 2.3.2.- FONDO DE ESCRITORIO.

Tanto el fondo de escritorio, como los colores primario y secundario del fondo se indican en *Gconf*.

*Gconf* es el "registro" de configuraciones de *Gnome*.

1 http://www.gnome.org/learn/admin-guide/2.14/

2 http://www.gnome.org/projects/gconf/

En este registro se guarda que archivo se cogerá como imagen de fondo del escritorio (entre otras muchas cosas). Y lo hace concretamente en la clave "/desktop/gnome/background/picture_filename", que para Guadalinex V3 tiene el valor "/usr/share/backgrounds/guadalinex_v3_bg.jpg". Podemos ver los valores relacionados con el fondo con el siguiente comando:

```
$ gconftool-2 -R /desktop/gnome/background
 color_shading_type = solid
 secondary_color = #2c160a
 primary_color = #2c160a
 picture_filename = /usr/share/backgrounds/guadalinex_v3_bg.jpg
 picture_options = stretched
 picture_opacity = 100
 draw_background = true
```

Dichos valores podemos cambiarlos para todo el sistema de la siguiente forma:

```
$ gconftool-2 --direct --config-source xml:readwrite:/etc/gconf/gconf.xml.defaults --type string --set /desktop/gnome/background/picture_filename "/usr/share/pixmaps/splash/backgrounds/mi_distro.jpg"
```

## 2.3.3.- IMAGEN "SPLASH" DEL INICIO DE SESIÓN

La imagen "splash" es la que aparece en el inicio de la sesión de Gnome, mientras se van cargando los diferentes elementos del escritorio.

El predeterminado se encuentra aquí:

```
/usr/share/pixmaps/splash/gnome-splash.png
```

Cambiar dicho "splash" puede hacerse cambiando directamente este archivo, o cambiando dicho valor en el "gconf".

## 2.3.4.- MENÚS.

Podemos encontrar información más completa sobre en el proyecto "Freedesktop"[1], así que aquí nombraremos los archivos y directorios más importantes:

Archivo en donde se define el menú de aplicaciones:

```
/etc/xdg/menus/applications.menu
```

Ejemplo de entrada para un submenú:

```
<!-- Education -->
<Menu>
  <Name>Education</Name>
  <Directory>Education.directory</Directory>
  <Include>
    <And>
      <Category>Education</Category>
    </And>
  </Include>
</Menu> <!-- End Education -->
```

1 http://standards.freedesktop.org/menu-spec/latest/

Directorio donde se guardan los ".directory", responsables de la apariencia y configuración de las entradas del menú:

```
/usr/share/desktop-directories/
```

Ejemplo de un ".directory":

*education.directory*

```
[Desktop Entry]
Encoding=UTF-8
Name=Education
Name[es]=Educación
Icon=package_edutaiment
Type=Directory
```

Los ".directory" son los que configuran la apariencia (icono, nombre que aparece, traducciones, etc) y configuración de un menú o submenú, pero para las entradas de las aplicaciones se usa el ".desktop", que es el mismo que sirve para poner "lanzadores" de aplicaciones en el escritorio o el panel.

Estos archivos podremos encontrarlos en el directorio "/usr/share/applications/" y podremos encontrar mas documentación sobre su formato y uso en la especificación oficial[1].

1 http://standards.freedesktop.org/desktop-entry-spec/latest/

Pero mostraremos un ejemplo, de todas formas:

*tomboy.dektop*

```
[Desktop Entry]
Encoding=UTF-8
Name=Tomboy
Name[es]=Notas de escritorio (Tomboy)
Comment[es]=Escriba notas en el escritorio
Exec=tomboy
Icon=tomboy
Terminal=false
StartupNotify=true
Type=Application
Categories=Application;Utility;
```

## 2.3.5.- PANELES

Para configurar los paneles de Gnome según nuestras necesidades, debemos configurarlos para un usuario de prueba y *capturar*, dicha configuración. Esto cogerá tanto los *applets*, como su posición, el tamaño, color y demás propiedades del panel.

Una vez tenemos configurado el panel a nuestro gusto, solo tenemos que volcar la información a un archivo:

```
$ gconftool-2 --dump /apps/panel > /usr/share/gnome/panel.entries
```

Para restaurarlo en la instalación del correspondiente *metapaquete de configuración*:

```
$ gconftool-2 --direct --config-source xml:readwrite:/etc/gconf/gconf.xml.defaults --load /usr/share/gnome/panel.entries
```

## 2.3.6.- TEMAS

También pueden cambiarse los temas mediante la configuración de Gnome:

```
$ gconftool-2 --direct --config-source xml:readwrite:/etc/gconf/gconf.xml.defaults --type string --set /apps/metacity/general/theme "Guadalinex"
$ gconftool-2 --direct --config-source xml:readwrite:/etc/gconf/gconf.xml.defaults --type string --set /desktop/gnome/interface/gtk_theme "Guadalinex"
$ gconftool-2 --direct --config-source xml:readwrite:/etc/gconf/gconf.xml.defaults --type string --set /desktop/gnome/interface/icon_theme "Guadalinex"
```

Eso sí, debemos asegurarnos de que exista el tema seleccionado en:

```
/usr/share/icons/[TEMA]
/usr/share/themes/[TEMA]
```

Tema de iconos y tema de Gtk respectivamente. El del gestor de ventanas ("Metacity") se debería encontrar en:

```
/usr/share/themes/[TEMA]/metacity-1/metacity-theme-1.xml
```

Así como sucedía con el tema para el GDM, no vamos a explicar cómo se crearía o modificarían estos temas pero puede consultarse aquí:

```
http://live.gnome.org/GnomeArt/Tutorials/GtkThemes
http://live.gnome.org/GnomeArt/Tutorials/IconThemes
http://developer.gnome.org/doc/tutorials/metacity/metacity-themes.html
```

## 2.3.7.- MOZILLA FIREFOX

Aunque el navegador "Firefox" no es parte de Gnome, *sí es parte del escritorio y un elemento básico dentro de él.* Los elementos más comunes que se pueden personalizar son los siguientes:

```
/etc/mozilla-firefox/pref/firefox.js
/etc/mozilla-firefox/profile
```

# 2.4.- AÑADIR Y QUITAR APLICACIONES.

A la hora de añadir o quitar aplicaciones de una distribución, es importante tener claro qué aplicaciones se desea (des)instalar en el sistema y qué repercusiones puede tener en relación con otras aplicaciones instaladas. En un sistema Debian/Ubuntu (y otras distribuciones), la instalación y desinstalación de aplicaciones se lleva a cabo a través de paquetes. Es importante conocer adecuadamente cómo funciona el sistema de paquete de Debian (DPKG) para gestionar adecuadamente las aplicaciones que van a formar parte del sistema final. En particular, es importante tener en cuenta las dependencias y conflictos entre aplicaciones, de manera que la (des)instalación de una aplicación determinada no afecte al buen funcionamiento de la distribución final.

Se van a señalar a continuación algunas pautas que deben seguirse a la hora de gestionar paquetes para conformar la distribución final y minimizar riesgos:

- Se debe utilizar una distribución base lo más estable posible, como es el caso de Debian “Sarge” o Ubuntu “Breezy”. Utilizar distribuciones inestables (Debian “Sid” o, en el momento de escribir este documento (10/05/06), Ubuntu “Dapper”) puede provocar errores, problemas e imprevistos varios no deseados.

- A la hora de seleccionar una aplicación para su inclusión en la distribución, hay que controlar en todo momento que las dependencias de la misma también se instalan. Además, hay que confiar en el sistema de dependencias de los paquetes de Debian/Ubuntu de manera que nos preocupemos de que se instale el “paquete padre” (todas sus dependencias se instalarán automáticamente gracias a “apt”). Es decir, cuando se desea disponer de un conjunto concreto de aplicaciones, hay que seleccionar el menor conjunto de paquetes que garantice el disponer de las aplicaciones deseadas (y no preocuparse por instalar manualmente las librerías (u otros paquetes) de los que dependen; esto lo gestionará automáticamente el sistema de dependencias de Debian).

- Cuando se desea eliminar una aplicación de la distribución hay que asegurarse, por una parte, de que no provoca la desinstalación de otros paquetes que dependen de ella y que puedan ser importantes para el resto

del sistema. Por otra parte, es conveniente asegurarse de que no quedan paquetes huérfanos[1] en el sistema tras la desinstalación de una aplicación. Para ello, es conveniente utilizar "deborphan", una herramienta que permite detectar paquetes huérfanos en el sistema.

## 2.5.- CONFIGURACIÓN DE APLICACIONES.

Una vez se ha seleccionado un conjunto concreto de aplicaciones para el sistema final, el siguiente paso consiste en constatar que el funcionamiento y la configuración de las mismas son los adecuados.

Para ello, lo más recomendable es disponer de un sistema base de pruebas y, sobre él, ir instalando una a una las aplicaciones deseadas. Con cada una de ellas, conviene llevar a cabo un proceso de validación y pruebas para garantizar que la aplicación funciona como se espera. Esto dependerá de la aplicación concreta. Por ejemplo, si se trata de una aplicación de escritorio (pongamos un navegador web), hay que garantizar que la entrada correspondiente del menú de aplicaciones se crea correctamente, que la aplicación se lanza en el idioma adecuado, que el directorio para guardar las descargas es el que se desea, que la página configurada como inicio es correcta, etc. Si no es el caso, habrá que analizar los ficheros de configuración de que consta la aplicación para saber qué y cómo llevar a cabo la personalización con respecto de la configuración por defecto de la misma.

Una vez se ha constatado que la aplicación funciona de acuerdo a nuestras necesidades hay que acotar los ficheros de configuración que se han modificado. Esto será imprescindible para continuar con el proceso de generación de la distribución final. Cómo proceder a partir de este momento dependerá del sistema de generación que se esté utilizando. En el caso de Guadalinex v3, se describe en el apartado Personalización de la distribución.

1 Un paquete *huérfano* es un paquete que se instaló en el sistema porque otro paquete dependía de él y que sigue existiendo en el sistema cuando se desinstala su paquete "padre". Éste es el caso, normalmente, de librerías necesarias para que una aplicación funcione. Cuando esta aplicación se desinstala y ningún otro paquete depende de la librería anterior se dice que se ha quedado huérfanas.

# 3.- SISTEMA NO LIVE

## 3.1.- OBTENER TODO LO NECESARIO

### 3.1.1.- DOS SABORES

El sistema de generación se puede obtener de dos formas distintas:

- Un ".tar" de un sistema ya configurado con idea de trabajar con el en "chroot".
- Montando el sistema de generación partiendo del código fuente existente en el subversion

```
svn co http://ws314.juntadeandalucia.es/guadalinex2005/trunk/generation_system/no_live/cdimage/
```

### 3.1.2.- ¿CÓMO SE HA GENERADO EL .TAR?

A continuación se listan los pasos seguidos para obtener, partiendo del contenido del repositorio de guadalinex, un sistema de generación listo para usar.

- Genera un directorio "guadalinex-v3/" con un sistema base:

```
debootsrap breezy guadalinex-v3/ http://es.archive.ubuntu.com/ubuntu
```

- Hacemos "chroot":

```
sudo chroot guadalinex-v3
```

- Ponemos a punto el sistema mediante configuraciones varias:

```
echo "es_ES.UTF-8 UTF-8" > /etc/locale.gen

locale-gen

echo "127.0.0.1       localhost       $HOSTNAME" > /etc/hosts
```

- Instalación de las dependencias:

```
apt-get install subversion debootstrap devscripts build-essential python bc python-apt mkisofs g++-3.4 fakeroot
```

- Añadimos un usuario y le damos permisos:

```
adduser guada

echo "guada   ALL=(ALL) ALL" >> /etc/sudoers

su guada
```

- Realizamos un “checkout” de la rama trunk del subversion de guadalinex

```
svn co http://ws314.juntadeandalucia.es/guadalinex2005/trunk
```

- Nos desplazamos a “generation_system/no_live/cdimage/”

- Rellenamos “cdimage/ftp/” con:

```
dists/

indices/

pool/
```

El contenido de dichos directorios puede obtenerse desde un CD.

- Se configuran las rutas absolutas en:

```
cdimage/uda/conf/d-i_flamenco.conf

cdimage/uda/conf/flamenco.conf
```

- Generación de un nuevo paquete “ubuntu-keyring”.

Es necesario tener un clave GPG para firmar el “Release.gpg” del repositorio que se incluye en el CD. Por ello, se ha generado un par de claves pública/privada con nombre "Guadalinex derivada" y fingerprint: CADFFC1A

A esta clave generada se le ha importado las claves contenidas en “ubuntu-keyring” (gpg --import fichero.gpg) para así obtener un anillo de confianza más amplio, es decir, con las claves importadas, tenemos confianza sobre el repositorio de Ubuntu y sobre el repositorio de la Junta de Andalucía.

Sustituimos la clave contenida en el paquete ubuntu-keyring por la clave pública generada:

```
cp .gpg/pubring.gpg  ubuntu-keyring-2005.01.12.1/keyrings/ubuntu-archive-keyring.gpg
```

Compilamos el paquete:

```
cd

cd ubuntu-keyring-2005.01.12.1

debuild -uc -us
```

Los paquetes obtenidos ".udeb" y ".deb" sustituyen a los contenidos en "cdimage/ftp/pool/main/u/ubuntu-keyring/". Tras incluir los nuevos paquetes, es necesario cambiar:

- El fingerprint que está configurado en "cdimage/etc/config"
- La variable "SIGNING_KEYID" contenida en "cdimage/debian-cd/CONF.sh"

A continuación, copiamos el nuevo par de claves al directorio "cdimage/secret/" tal que:

```
secret/

`-- dot-gnupg

   |-- pubring.gpg

   `-- secring.gpg
```

# 3.2.- ARQUITECTURA DEL SISTEMA DE GENERACIÓN

El sistema de generación más actual se encuentra bajo el repositorio de Guadalinex en el directorio "trunk/generation_system/no_live/cdimage", contenidos bajo dicho directorio nos encontramos con:

bin/ (Ejecutables)

debian-cd/ (Contiene el ~80% de la lógica de todo el sistema de generación)

etc/ (Configuraciones varias)

ftp/ (Contenidos de paquetes, documentación e imagenes)

germinate/ (Aplicación encargada de generar listas)

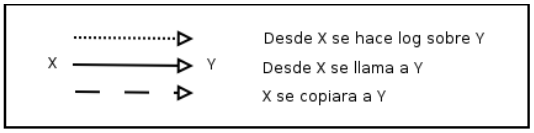
log/ (Registro de información)

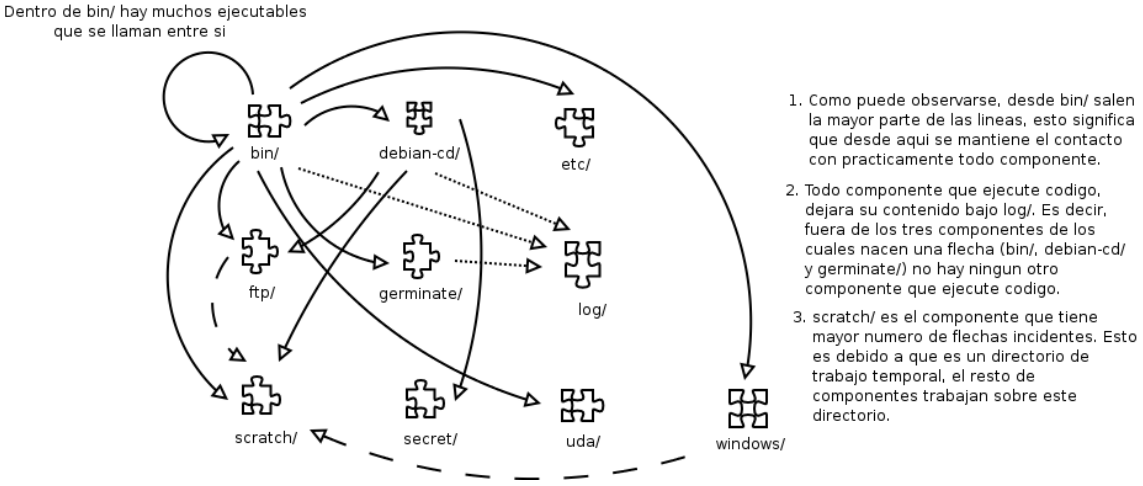
scratch/ (El directorio temporal)

secret/ (Contiene la clave GPG)

uda/ (UDA, Unión de Distribuciones Autónomas)

windows/ (Datos de windows)




A continuación, se describirán los directorios listados anteriormente.

### 3.2.1.- BIN/

Directorio que contiene scripts varios, cabe destacar la existencia de un ejecutable llamado “generate_no-live.sh”, dicho ejecutable es el padre de todos los demás scripts útiles.

Muchos de los scripts contenidos en este directorio ya no son usados, nótese que Guadalinex v3 está basada en Ubuntu Breezy, cuando comenzó el desarrollo de Guadalinex, Breezy aún no era estable y por tanto era necesario mantener sincronizados el constante flujo de paquetes que había en el repositorio de pobrezas. Por este motivo hay varios ejecutables con fines de sincronización de paquetes que ya no son usados.

```
bin/

.

|-- README

|-- compile-conf

|-- diff-tasks

|-- find-mirror

|-- functions.sh

|-- generate_no-live.sh    <--- Script "padre"

|-- germinate-to-tasks

|-- list-seeds

|-- next-build-date

|-- not-used      (Scripts no usados)

|   |-- anonftpsync

|   |-- build-image-set

|   |-- ...

|-- run-germinate

|-- update-debs

|-- update-dist

|-- update-indices

|-- update-installer-i386

|-- update-local-indices

|-- update-pool

|-- update-tasks

|-- update-udebs
```

## GENERATE_NO-LIVE.SH

Es el encargado de gestionar todo el proceso, obtiene las listas necesarias, las convierte en tareas, actualiza los datos del repositorio local y da paso a debian-cd.

## UPDATE-UDEBS

Dentro de este fichero hay una variable “MIRROR”. Si dispone de un mirror local, cambie esta variable. Al ejecutar este script, se descargaran todos los “.udeb” necesarios y se actualizarán los ya existentes. Todos los “.udeb” se irán colocando en “ftp/pool/”

## UPDATE-DEBS

Dentro de este fichero hay una variable MIRROR. Si dispone de un mirror local, cambie esta variable.

Al ejecutar este script, se descargaran todos los “.deb” necesarios excluyendo los listados en “uda/lists/exclude”

Todos los “.deb” se irán colocando en “ftp/pool/”

## UPDATE-INSTALLER-I386

Sincroniza por rsync el directorio “ftp/dists/breezy/main/installer-i386”

## UPDATE-INDICES

Sincroniza por rsync el directorio “ftp/indices”

## UPDATE-POOL

Enmascara a todos los scripts anteriores:

```
update-pool --all            (Actualiza todo)

update-pool --udebs --debs   (Únicamente actualiza los paquetes)

update-pool --indices        (Únicamente actualiza indices)
```

### 3.2.2.- DEBIAN-CD/

Conjunto de herramientas para generar CDs instalables basados en “debian-installer”.

### 3.2.3.- ETC/

Bajo este directorio se encuentra el fichero “config”, en el cual se definen variables varias: nombre del proyecto, “keyid”, ...

### 3.2.4.- FTP/

En este directorio es donde se encuentran los paquetes, la documentación y las imágenes a incluir. Ha de ser rellenado con:

```
dists/
pool/
indices/
```

### 3.2.5.- GERMINATE/

Consiste en un programa en python que obtiene listas de paquetes, la lógica es compleja pero es muy útil para generar las listas de forma jerárquica. Al pasarle varios paquetes, “germinate” genera una serie de listas en función de dichos paquetes que serán útiles para generar tareas posteriormente. Aunque podría prescindir de él, es recomendable usarlo porque ayuda en la gestión de dependencias.

### 3.2.6.- LOG/

En este directorio se escribe el registro de la última generación.

### 3.2.7.- SCRATCH/

Este directorio es de contenido variable:

```
scratch/
  guadalinex/
    apt/
    debian-cd/
      i386/
        ____.list
        MD5SUMS
        ____.iso
    debootstrap/
    germinate/
    tasks/
    tmp/
```

### 3.2.8.- SECRET/

Este directorio está contenida la clave GPG usada para firmar el Release del repositorio del cd.

```
secret/
 `-- dot-gnupg
     |-- pubring.gpg
     `-- secring.gpg
```

### 3.2.9.- UDA/

Bajo este directorio, nos encontramos con cuatro subdirectorios que intentan facilitar algunas tareas a la

hora de realizar cambios.

```
conf/
isolinux/
lists/
pkgs/
```

## CONF/

En este directorio se podrán encontrar diversos ficheros .conf usados por bin/update-dist para generar ficheros varios (Release, Release.gpg, Packages.gz) que estarán contenidos en ftp/dists/

## ISOLINUX/

Todos los archivos contenidos en este directorio sustituirán a los generados por debian-cd.

## LISTS/

En este directorio hay tres listas bien diferenciadas:

- exclude

  Todo paquete en esta lista no se incluirán en el cd. Nótese que el sistema está basado en los CDs de breezy, esta lista será un listado de aplicaciones que se incluyen en el cd de breezy pero que no se incluirán en el cd de una distribución uda.

- include_breezy

  Está lista será incluida en el cd, esta lista estará contenida por todo paquete que se encuentre en el repositorio de breezy pero que no se encuentren en el cd de breezy por defecto.

- include_uda

  También será incluida en el cd esta lista que está contenida por todo paquete que no se encuentre en el repositorio de breezy.

## PKGS/

Este directorio es de vital importancia. Al incluir un .deb en este directorio, bin/upgrader (lanzado desde generate_no-live.sh) lo copiará al pool/ si no existe ya, o en el caso de que existiera se copiaría si fuera una versión más actual quitando del pool/ la versión más antigua. Sólo se copiará si el paquete es mayor estricto que la versión existente en el pool.

### 3.2.10.- WINDOWS/

Bajo este directorio está contenido una serie de datos y un script que lanza un navegador con ayuda sobre guadalinex en windows.

## 3.3.- GENERACIÓN

Primero, en un chroot, colocamos al usuario correcto, en el lugar indicado:

```
sudo chroot guadalinex-v3

su guada

cd ~/no_live/cdimage/
```

Podría hacerse con el usuario root, pero por motivos de seguridad es preferible trabajar con un usuario sin permisos de administración.

Posteriormente, desde el directorio ".../cdimage/" realizamos la carga del archivo "poner_a_punto".

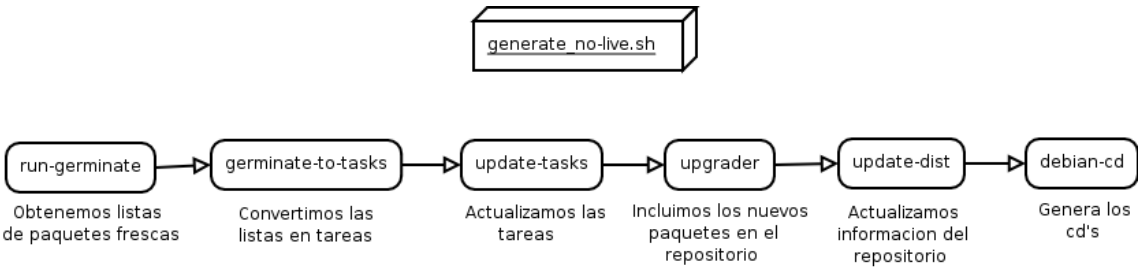
```
. poner_a_punto
```

Ejecutamos el comando nirvana:

```
generate_no-live.sh
```

Recogemos la iso que se encuentra en "/scratch/guadalinex/debian-cd/i386"

¿Qué es lo que ha ocurrido? "generate_no-live.sh" ha llamado a una serie de ejecutables que ponen todo a punto para que "debian-cd" genere una "iso" correcta.




# 3.4.- EL INSTALADOR

La instalación de Guadalinex se compone de dos fases bien diferenciadas:

- Primera fase: "Debian installer"
- Segunda fase: "base-config"

El código fuente más actual de los paquetes del instalador de Guadalinex puede encontrarse en el subversion bajo el directorio: "/guadalinex2005/trunk/no_live_installer/default/src".

## 3.4.1.- DEBIAN INSTALLER

La primera fase de la instalación es gobernada por "debian installer" con algunas modificaciones realizadas por Ubuntu a las que a su vez, se le han realizado algunas modificaciones para Guadalinex. Es común ver escrito "debian installer" como "d-i" y también es común ver referencias a "ubuntu installer" (u-i).


La principal virtud de “debian installer” es su gran modularidad. Todo en “debian installer” es un “udeb” (oficialmente se pronuncia micro-deb). Prácticamente, es igual que un paquete debian estándar pero se le permite infringir la política de debian en ciertos puntos para que el paquete sea más ligero. A partir de ahora cuando hablemos de un módulo en “debian installer”, estaremos hablando de un “udeb” y viceversa.

A continuación, se darán unas nociones básicas sobre los principales roles dentro del instalador:

- debconf
- main-menu
- anna
- discover

## DEBCONF

Toda configuración de un paquete se hace a través de “debconf”, esta capa es de vital importancia.

Cada configuración tiene:

- Una opción por defecto configurable.
- Una prioridad, dicha prioridad puede tener valor: low, medium, high y critical. Si la prioridad del sistema es critical, sólo se mostrarán las preguntas de “debconf” con prioridad de critical, si la prioridad del sistema es high, se mostrarán las preguntas de “debconf” high y critical... Es decir, según la prioridad del sistema, se mostrarán las preguntas con dicha prioridad y todas las de mayor relevancia. El instalador de Guadalinex tiene la prioridad del sistema a critical.

Gracias al sistema de prioridades podemos facilitar la vida al usuario evitando preguntarle configuraciones.

Otro punto de que posee un gran potencial es su variedad de “frontends”: texto, ncurses, gtk (en desarrollo), ...

Como consecuencia de este sistema, podemos pasar al instalador un fichero de texto plano (llamado “preseed”) con configuraciones varias para distintos programas, Guadalinex tiene dos “preseed” que se cargan en función de los parámetros de arranque. Se encuentran en “cdimage/debian-cd/data/breezy/preseed” con nombres “uda.seed” y “uda-check.seed”, este último es el que se carga al arrancar el instalador de Guadalinex en modo "testcd". Únicamente difieren en esta linea:

```
d-i cdrom-checker/start boolean true
```

El fichero uda.seed es el preseed que se carga por defecto, algunos ejemplos:

```
base-config tzconfig/choose_country_zone_multiple string Europe/Madrid (mainland)

base-config base-config/install-language-support boolean true

xserver-xorg xserver-xorg/config/inputdevice/keyboard/layout string es
```

La primera linea selecciona la zona horaria, la segunda hace que se instale el soporte de lenguajes y la tercera linea selecciona por defecto el teclado español.

## MAIN-MENU

La importancia de este elemento en la primera fase de la instalación reside en que es el gestor de todo módulo cargado en “debian installer”. Cada módulo tiene una dependencia, por ejemplo, no podemos configurar grub si aún no se ha desempaquetado el sistema base. main-menu se encarga de gestionar que módulo es accesible en cada momento según sus dependencias y da paso al siguiente módulo según su prioridad.

## ANNA

Su significado es: Anna's Not Nearly Apt, es el APT (Advanced Packagement Tool) de “debian installer” enfocado a ser mucho más ligero y con únicamente las funciones necesarias para el instalador.

## DISCOVER

Por último, mencionar una pieza clave de “debian installer” que se encarga de la autodetección de hardware, “discover” reconoce y propone a través de “debconf” configuraciones según el hardware detectado.

### 3.4.2.- BASE-CONFIG

En el primer arranque tras la primera fase de la instalación, “base-config” tomará el control del sistema y hará las configuraciones finales.

El fichero “pkgsel” contenido en “guadalinex2005/trunk/no_live_installer/default/src/base-config/base-config-2.67uda/lib/menu/” dirige la mayor parte de la segunda fase de la instalación, dentro de este ejecutable, se llama a su vez a varios componentes:

- uda-postinstall

  Se encargada de mostrar la barra de progreso de la instalación.

- uda-postconfig

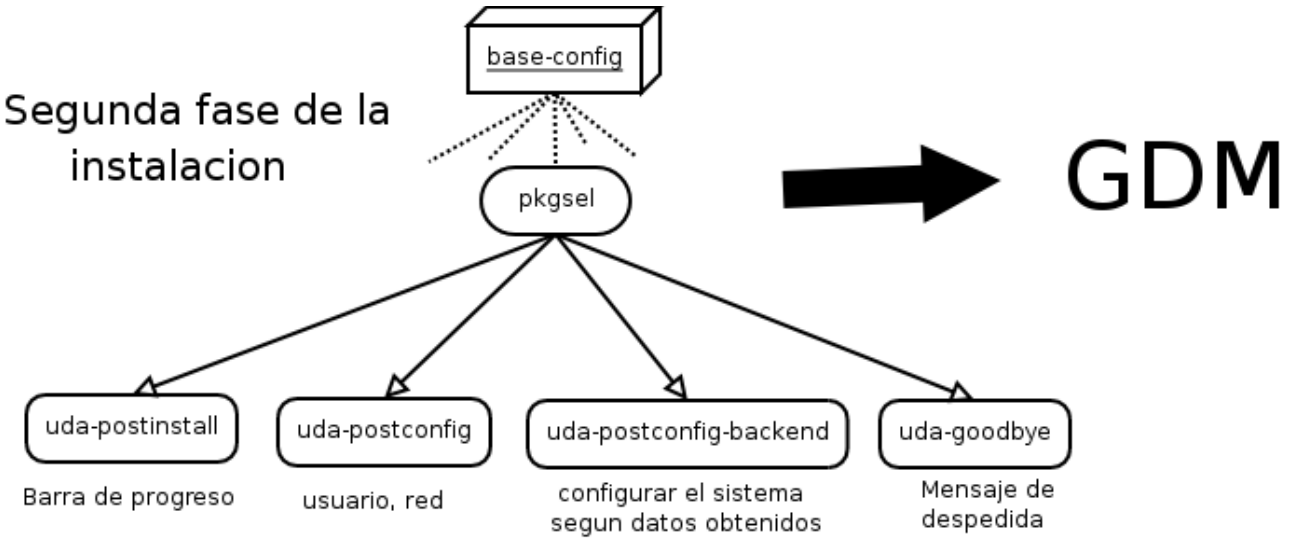
  Se encarga de preguntar datos al usuario.

- uda-postconfig-backend

  Se encarga de dar de alta el usuario, configurar la red, poner los repositorios, ...

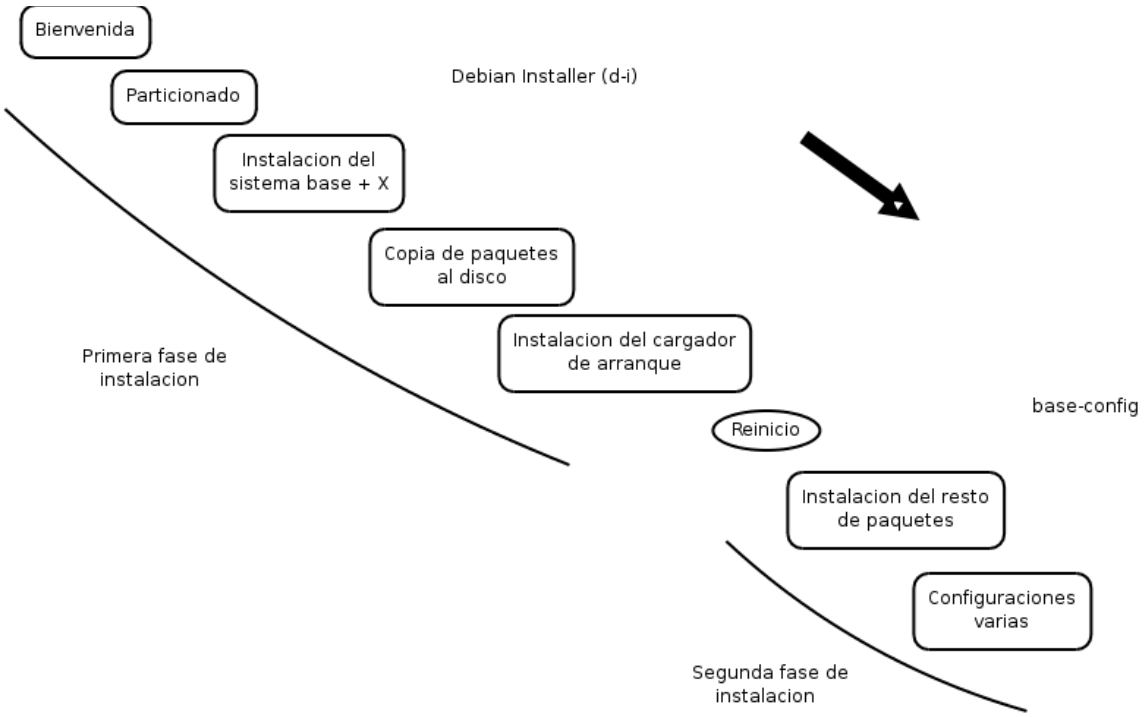
- uda-goodbye

  Muestra un mensaje de despedida.




## 3.4.3.- FASES DE LA INSTALACIÓN

- bienvenida
- particionado
- instalación de los paquetes más básicos
- copia de paquetes al disco
- instalación del cargador de arranque
- instalación del resto de paquetes
- Alta de usuario




# 3.5.- BRANDING

## 3.5.1.- CAMBIAR MENSAJE DE BIENVENIDA: UDA-FIRST-MESSAGE

"uda-first-message" es un módulo desarrollado por Guadalinex para mostrar un mensaje al inicio de la instalación. El código fuente de este "udeb" se puede encontrar en "guadalinex2005/trunk/no_live_installer/default/src/uda-first-message/uda-first-message-0.05/" Una vez en el directorio indicado debemos editar el "template" que se encuentra en el directorio "debian/" con el nombre uda-first-message.templates.

Tras realizar las modificaciones pertinentes recompilamos el paquete:

```
debuild -uc -us
```

A continuación hemos de insertar este "udeb" dentro del "initrd" porque al ser uno de los primeros "udebs" que se cargan en el sistema, es necesario que se encuentre en este componente. Por tanto procedemos a generar un nuevo "initrd" con este "udeb", para ello usaremos un paquete llamado "debian-installer".

Nos desplazamos al directorio “guadalinex2005/trunk/generation_system/no_live/debian-installer/debian-installer-20050317ubuntu19/build y vemos el contenido de directorio localudebs/”

```
$ ls localudebs/

main-menu_1.07ubuntu1_i386.udeb

uda-first-message_0.05_i386.udeb

uda-prepartman-message_0.01_i386.udeb
```

Como podemos observar, hay contenidos tres paquetes: main-menu, uda-first-message y uda-prepartman-message, dichos “udebs” han sido generados por y para Guadalinex para añadir mensajes extras durante el proceso de instalación.

A continuación debemos copiar el “.udeb” que habíamos generado en el directorio “localudebs/” y generar de nuevo isolinux, para ello ejecutaremos en el directorio “build/”

Cambiamos al directorio “build/” y ejecutamos:

```
dpkg-scanpackages . /dev/null > Packages
```

Así hemos regenerado las listas de paquetes locales, consecuentemente, el sistema sabrá que paquetes hay nuevos o modificados. Tras ello, ejecutamos:

```
fakeroot make build_cdrom_isolinux
```

Al finalizar, obtendremos un directorio “dest/” que a su vez, contiene un directorio “cdrom/” este último es el que necesitamos cambiar. Para ello, realizamos lo siguiente:

```
rm -rf ../../../cdimage/ftp/dists/breezy/main/installer-i386/current/images/cdrom/

cp -r dest/cdrom ../../../cdimage/ftp/dists/breezy/main/installer-i386/current/images/
```

Llegados a este punto, ya no es necesario tocar nada más, “debian-cd” se encargará de usar los datos que le hemos dejado para generar las imágenes de arranque correctamente.

## 3.5.2.- CAMBIAR IMÁGENES DE LA SEGUNDA FASE DE LA INSTALACIÓN: UDA-POSTINSTALL

Las imágenes que se muestran durante la segunda fase de la instalación están contenidas en “guadalinex2005/trunk/no_live_installer/default/src/uda-postinstall/uda-postinstall-0.21/src/branding/Guadalinex/es”, tras modificarlas, habría que regenerar el paquete e incluirlo en el repositorio, para ello, una vez en el directorio “uda-postinstall-0.21/” recompilamos las fuentes:

```
debuild -uc -us
```

Y sustituimos los paquetes contenidos en “pool/main/u/uda-postinstall/” por los generados: uda-postinstall_0.21-1_i386.deb y uda-postinstall-guadalinex-branding_0.21-1_i386.deb

## 3.5.3.- CAMBIAR ENTRADA DE GRUB

Si observamos los paquetes contenidos bajo “/guadalinex2005/trunk/no_live_installer/default/src” veremos que hay dos paquetes relacionados con el cargador de arranque: grub y grub-installer. La diferencia es que grub-installer es un “udeb” que en la primera fase de la instalación se encarga de preparar el disco para arrancar y, por tanto, nos permite acceder a la segunda fase mientras que grub, es el paquete .deb que se instala en la segunda fase de la instalación.

Es decir, si queremos cambiar la entrada de grub para que no se vea en el primer arranque la entrada original de Guadalinex es necesario cambiar grub-installer. Desde el directorio src/ ejecutamos el comando:

```
GUADA_FICHEROS=$(fgrep -r Guadalinex grub* | grep -v .svn | awk 'BEGIN { FS =
":" } { print $1 }')
```

Nos devuelve todos los ficheros que contengan Guadalinex. Podemos hacer la sustitución de la siguiente forma:

```
sed -i "s/Guadalinex/Guadalinex para peluqueros/g" $GUADA_FICHEROS
```

Tras ello, nos queda regenerar cada paquete y sustituir a los actuales, se encuentran en "cdimage/ftp/pool/main/g/grub y cdimage/ftp/pool/main/g/grub-installer".

### 3.5.4.- APARICIONES DE GUADALINEX EN DEBIAN INSTALLER

Estamos ante un caso idéntico al anterior apartado, nos desplazamos al código fuente de main-menu: "guadalinex2005/trunk/no_live_installer/default/src/main-menu/main-menu-1.07ubuntu1" y ejecutamos:

```
sed -i "s/Guadalinex/Guadalinex para ella/g" $(fgrep -r Guadalinex * | grep -v .svn)
```

Tras ello, recompilamos el "udeb" y habría que incluirlo en isolinux, por tanto habría que seguir los pasos descritos en "Cambiar mensaje de bienvenida: uda-first-message". También apariciones de Guadalinex en partman, el procedimiento sería equivalente.

## 3.6.- EJEMPLOS PRÁCTICOS

### 3.6.1.- INCLUIR UN NUEVO PAQUETE

Se pueden dar dos casos:

1. Cambiar un paquete cuyo nombre y versión es igual que la que hay en el "pool/"

   En este caso, lo más fácil es hacerlo a mano, para ello simplemente colocaremos el paquete en su

directorio correspondiente bajo “pool/main/{carácter}” o “pool/main/lib{carácter}”, siendo carácter la inicial correspondiente del paquete fuente. Nótese que la estructura correcta distingue los directorios donde están contenidas las librerías (lib{carácter}) con el resto de paquetes ({carácter}). El directorio donde debe estar contenido un paquete debe ser: “ftp/main/{carácter}|lib{carácter}/{nombre del paquete fuente}/”.

Por ejemplo, el paquete binario “mount” tiene como paquete fuente “util-linux”, por tanto debería encontrarse en “ftp/main/u/util-linux”.

2. Incluir un nuevo paquete o incluir una versión más nueva de la que hay actualmente. Para ello, se puede hacer a mano o de una forma un poco más cómoda que es colocar el paquete deseado en “guadalinex2005/trunk/generation_system/no_live/cdimage/uda/pkgs”, ya que todo paquete contenido bajo ese directorio será indexado y colocado correctamente en el repositorio.

NOTA IMPORTANTE: Para que un paquete sea incluido en el cd:

- el paquete ha de estar en “pkgs/”

  ó

- debe estar en el “pool/” y a la vez listado en uno de los dos archivos siguientes: “guadalinex2005/trunk/generation_system/no_live/cdimage/uda/lists/include_breezy” o “guadalinex2005/trunk/generation_system/no_live/cdimage/uda/lists/include_uda”

Tras los cambios realizados, ejecutaríamos “generate_no-live.sh”

## 3.6.2.- BORRAR UN PAQUETE

Antes de borrar un paquete debemos estar seguro de lo que estamos haciendo puesto que si algún paquete que vaya a instalarse depende de dicho paquete, la instalación fallará. El error que se obtendría sería un pantallazo en negro en la segunda fase de la instalación.

Si estamos seguros de que no vamos a romper las cosas, únicamente es necesario borrarlo del “pool/” aunque también sería recomendable borrarlo de la lista en la que estuviera contenido “cdimage/uda/lists/include_breezy” o “cdimage/uda/lists/include_uda”). Otra opción sería no borrar el paquete del

"pool/" sino que únicamente borrarlo de la lista, así el paquete no se incluiría en el cd.

### 3.6.3.- CAMBIAR EL TEXTO DE ISOLINUX

Para cambiar el texto de isolinux simplemente editamos el archivo "guadalinex2005/trunk/generation_system/no_live/cdimage/uda/isolinux/isolinux.txt".

Todos los archivos contenidos bajo "cdimage/uda/isolinux" se copiaran cada vez que se genere un nuevo cd, por tanto, cualquier cambio a la imagen (splash.rle) o a los textos de ayuda (f1.txt, f2.txt, ...) tendrán efecto inmediato.

### 3.6.4.- CAMBIAR LOS REPOSITORIOS

Los repositorios se colocan prácticamente al final de la segunda fase de la instalación. No es factible poner estas listas dentro de un paquete para que al desempaquetarlo se coloquen en su sitio correctamente (Por ejemplo, sources.list en "/etc/apt/") puesto que usamos apt para la instalación de los paquetes y si durante la instalación de un conjunto de paquetes se cambia el sources.list apt daría error.

Por tanto hay que cambiar los repositorios una vez terminada la instalación de todos los paquetes. Esto se hace en uda-postinstall, para ello, nos desplazamos al código fuente de dicha aplicación "guadalinex2005/trunk/no_live_installer/default/src/uda-postinstall/uda-postinstall-0.21", editamos "src/uda-postconfig-backend.sh", y cambiamos las siguientes lineas:

```
echo '#TITLE:Sitio principal de Guadalinex en la Junta de Andalucia' >> /etc/apt/sources.list.uda

echo '#ID:jda' >> /etc/apt/sources.list.uda

echo "deb http://repositorio.guadalinex.org/ubuntu-breezy breezy main restricted universe multiverse" >> /etc/apt/sources.list.uda

echo "deb http://repositorio.guadalinex.org/guadalinex-flamenco flamenco main" >> /etc/apt/sources.list.uda

echo "deb http://repositorio.guadalinex.org/guadalinex-flamenco flamenco-updates main restricted universe multiverse" >> /etc/apt/sources.list.uda

echo "deb http://repositorio.guadalinex.org/guadalinex-flamenco flamenco-security main restricted universe multiverse" >> /etc/apt/sources.list.uda

echo "deb http://repositorio.guadalinex.org/guadalinex-flamenco flamenco-backports main restricted universe multiverse" >> /etc/apt/sources.list.uda

echo '#END' >> /etc/apt/sources.list.uda
```

Acorde a nuestras necesidades, podríamos añadir nuevos repositorios o modificar lo ya existentes. En el mismo nivel que el ejecutable “uda-postconfig-backend.sh”, se encuentran los siguientes ficheros:

```
repositorio.guadalinex.org_guadalinex-flamenco_dists_flamenco-backports_main_binary-i386_Packages

repositorio.guadalinex.org_guadalinex-flamenco_dists_flamenco-backports_multiverse_binary-i386_Packages

repositorio.guadalinex.org_guadalinex-flamenco_dists_flamenco-backports_restricted_binary-i386_Packages

repositorio.guadalinex.org_guadalinex-flamenco_dists_flamenco-backports_universe_binary-i386_Packages

...
```

Dichos ficheros contienen la base de datos de paquetes que se actualiza al hacer un "apt-get update".

Coloquemos en un sistema debian cualquiera los repositorios que vayamos a usar en la distribución y actualizamos la lista de paquetes con "apt-get update". Tras ello, recogemos las listas descargadas que se encuentran en "/var/lib/apt/lists/" y las colocamos en el directorio "guadalinex2005/trunk/no_live_installer/default/src/uda-postinstall/uda-postinstall-0.21/src" sustituyendo o añadiendo según convenga.

Dentro del script ya mencionado "uda-postconfig-backend.sh" hay una linea que contiene:

```
mv /usr/share/uda-postinstall/backend/repositorio* /var/lib/apt/lists
```

Dicha linea tendría que ser eliminada en el caso de que no se usaran los repositorios de Guadalinex y habría que añadir una nueva que copie las listas añadidas. Estos archivos no pertenecen a un paquete (se ponen a fuego) como en toda distribución debian porque son datos variables.

### 3.6.5.- CAMBIAR LA CLAVE GPG

Es común que una derivada quiera tener su propia clave privada sin que nadie más la posea, por ello se describe como generar una nueva clave, recompilar los paquetes necesarios y los cambios que habría que hacer en el sistema de generación.

Con el paquete gnupg instalado ejecutamos:

```
gpg --gen-key
```

Elegimos todas las opciones por defecto y cuando pregunte la clave pulsaremos intro. Obtendremos en nuestro directorio “$HOME/.gnupg/” un par de claves pública (pubring.gpg) y privada (secring.gpg). Es importante recalcar la importancia de no traspasar el archivo secring.gpg a nadie.

Si ejecutamos “gpg $HOME/.gnupg/secring” obtendremos información de la clave, por ejemplo:

```
sec  1024D/5318812F 2004-11-19 Carlos Parra Camargo <carlospc@gmail.com>

uid                  Carlos Parra Camargo <cparra@emergya.info>
```

En este caso, el “KEYID” de la clave es 5318812F. El siguiente paso vendría por recompilar el paquete ubuntu-keyring con la nueva clave y configurar el sistema de generación. Estos pasos están descritos en la primera sección, en el apartado "¿Cómo se ha generado el .tar?".

# 4.- SISTEMA LIVE

## 4.1.- INITRAMFS: DONDE TODO EMPIEZA

### 4.1.1.- ¿QUÉ ES?

El “initramfs” es la versión moderna del antiguo “initrd”. Pero seguro que esto no nos dice gran cosa.

Y si decimos que es un sistema mínimo que se monta en la memoria, antes de montar el sistema (instalado o no) que se arrancará después, ya sabemos algo más. Aunque, probablemente, aún nos preguntemos quién lo monta, por qué, qué se hace ahí y que tiene todo esto que ver con los sistemas live.

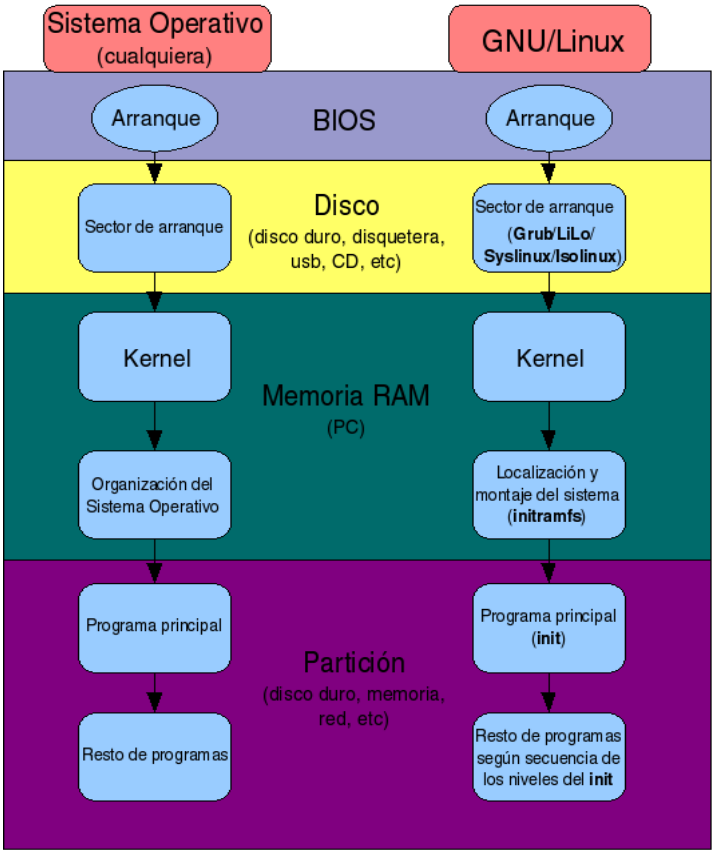



*Figura 9: Arranque de un sistema operativo cualquiera y el de GNU/Linux*

Pues bien, para aclararnos un poco, lo mejor es empezar por el principio: ¿Cómo arranca un sistema operativo y en concreto *GNU/Linux*?(Figura 9)

Según vemos en la Figura 9, una vez es cargado el kernel en memoria, junto con él se carga (si existe) el "initramfs", se lanza la ejecución del kernel y éste busca dónde debe arrancar la distribución. Si se carga un "initramfs", éste le servirá de sistema temporal desde el que detectar el disco donde está la distro y hacer más comprobaciones.

El "initramfs" es, en conclusión, un conjunto mínimo de directorios y archivos con los que el kernel, durante sus primaras fases de carga y ejecución, monta un pequeño sistema linux temporal, en la memoria, que le permite hacer detecciones de "hardware" básicas. Así como ejecutar pequeños programas o scripts.

En realidad es un pequeño sistema live que sirve al kernel para poder ser más flexible y soportar más hardware. También para lanzar programas tipo "usplash", para tener arranques gráficos sin tener que parchear el kernel.


*Figura 10: De las BIOS al "init"*

Una vez se localiza el sistema (instalado o live), se monta y se arranca su programa principal, es decir, el "init". El encargado de arrancar los sistemas GNU/Linux actuales.

La utilidad para un sistema live de un "initramfs" es obvia, el "initramfs" es el sistema live por excelencia. Ya lo tiene y hace todo. Sólo necesitamos pasarle una distro que arrancar y convencerle (4.2.3 Squashfs + Unionfs) de que es un sistema *instalado* y que lo puede arrancar. Esto es básicamente necesario para poder arrancar la distro completa, en memoria, y no sólo un sistema mínimo como es el "initramfs".

## 4.1.2.- ESTRUCTURA

Para Guadalinex v3 se creó un sistema live basado en el "initramfs" de Ubuntu y su estructura (Figura 11),

```
/
 |-- bin
 |-- conf
 |   |-- halt
 |   |-- initramfs.conf
 |   |-- install.desktop
 |   `-- modules
 |-- etc
 |-- init
 |-- lib
 |-- modules
 |-- sbin
 |-- scripts
 |   |-- functions
 |   |-- init-bottom
 |   |-- init-premount
 |   |   `-- acpid
 |   |-- init-top
 |   |   `-- usplash
 |   |-- live
 |   |-- live-bottom
 |   |   |-- adduser
 |   |   |-- ejectcd
 |   |   |-- fstab
 |   |   |-- hacks
 |   |   |-- hwdetect
 |   |   |-- init
 |   |   |-- log
 |   |   `-- md5
 |   |-- live-premount
 |   |   |-- local
 |   |   |-- setup_image
 |   |   `-- tmpfs
 `-- usr
     `-- lib
         `-- usplash
```

*Figura 11: Estructura del initramfs (simplificada)*

que añadía un tipo de arranque (**LIVE**) a los ya soportados por este sistema (**LOCAL**: para arrancar desde discos conectados localmente como discos duros IDE, SCSI, SATA, usb, firewire, etc; y **NFS**: para arrancar sistemas remotos vía LTSP).

El script "/init" (dentro del "initramfs") es el primero en ejecutarse al cargarse el "initramfs" y el que se lleva el peso y control de todo lo que pasa hasta que se cambia al sistema final (instalado o live).

Y su forma de funcionar es análoga al propio "init" de un sistema GNU/Linux normal. Es decir, va ejecutando scripts por niveles, estando éstos distribuidos en directorios.

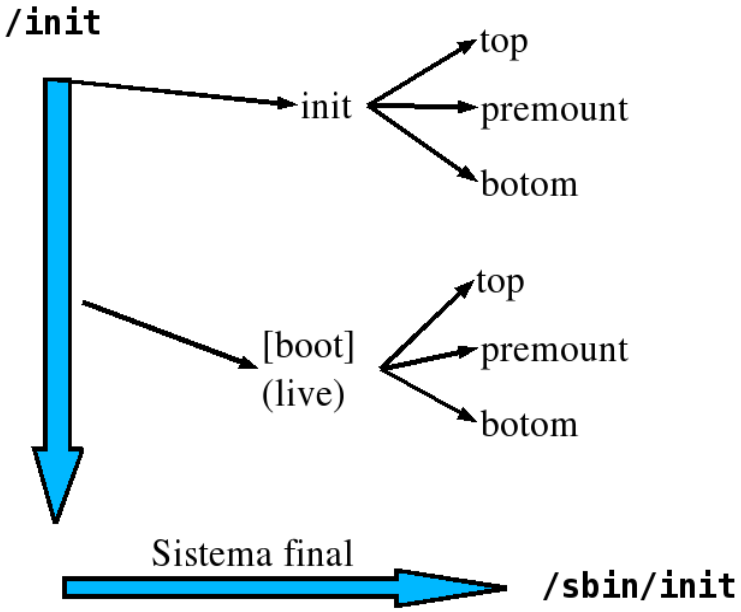



*Figura 12: Secuencia de ejecución del "init", dentro del "initramfs"*

## 4.1.3.- ARCHIVOS PRINCIPALES

Hagamos un repaso de los archivos principales del “initramfs”. Unos son importante porque realizan tareas importantes y otros porque son los que podremos usar para personalizaciones.

- ***/etc/mkinitramfs/initramfs.conf*** : Es el archivo de configuración del programa “mkinitramfs” que genera el archivo “initramfs”. Hay que tener en cuenta que este archivo esta ubicado en el sistema desde el que se ejecuta el “mkinitramfs”, generalmente desde el que se genera la distribución, no desde la propia distribución que se está generando.
- ***[initramfs]/init*** : Este es el script principal dentro del “initramfs”. Lleva el control de la ejecución dentro del “initramfs”.
- ***[initramfs]/scripts/live*** : Script donde se monta el sistema live con “unionfs”. Aquí está la función “mountroot”, llamada desde el “/init” y encargada del establecer el punto de montaje donde estará el

sistema final.

- ***[initramfs]/scripts/live-premount/*** :
  - ***local*** : Este script se encarga de localizar la distribución en CDs, discos locales, conectados por USB, etc. Busca el archivo "META.squashfs"
  - ***setup*** : Monta la imagen comprimida de la distribución, para poder ser usada posteriormente.
- ***[initramfs]/scripts/live-bottom/*** :
  - ***init*** : Aquí se cambia lo necesario del sistema de script de inicio de la distribución para evitar ejecutar script no necesarios o peligrosos en un sistema live.
  - ***hwdetect*** : Es el encargado de detectar y configurar el hardware de la máquina en la que se arranque.
  - ***hacks*** : Sirve para meter soluciones a bugs temporales, o cosas específicas de nuestra distribución. Cualquier cosa que no cuadre en ningún otro script, que sea temporal o muy específico, deberíamos ponerlo aquí.
- ***[initramfs]/tmp/initramfs.debug -> /var/log/initramfs.debug*** : Este es el archivo donde se guarda (si se le pasa al arranque la opción "debug") la información de depuración de los scripts. Aquí podremos encontrar alguna pequeña ayuda a entender lo que pasa, en case de fallar algo en el arranque.

# 4.2.- SQUASHFS Y UNIONFS: MÁS EN MENOS

## 4.2.1.- SQUASHFS

*Squashfs* es un sistema de ficheros comprimido. Bien, pero... ¿y eso qué significa?

Vayamos por partes. Un **sistema de ficheros**, *grosso modo*, es una relación entre el espacio un disco y la información que contiene. Eso, más información sobre la posición de cada dato, su relación con otros datos y un sistema de permisos. Es lo que hace que lo que en realidad es un montón de *bytes* agrupados en *clusters*, *sectores* y demás, tanto el sistema operativo, como los programas y nosotros mismos, lo vemos como archivos, directorios y espacio libre.

Pero no es más que una manera de ver esa información, una traducción a un aspecto más entendible por nosotros.

Sabiendo esto, podemos decir, que un sistema de ficheros comprimido, es aquel cuya información esta comprimida, pero que nosotros la vemos como si no lo estuviera. Es decir, la traducción ya no es de *clusters* y demás, sino de porciones de información comprimida a archivos, directorios, etc.

Este proceso es relativamente rápido y fácil de hacer. El de leer datos de un sistema comprimido. Pero escribir, ya es más complejo, ya que implicaría reorganizar toda la información, hacer las modificaciones pertinentes y volver a comprimir. Por esta razón generalmente los sistemas de este tipo, son sistema de *sólo lectura*. Esto quiere decir, que se puede leer, pero no se puede escribir en ellos.

Con toda esta introducción teórica lo que queremos decir es que el *squashfs* es un sistema de ficheros con el que podemos guardar mucha información en poco espacio, al guardarse de forma comprimida. Pero que esa información, una vez guardada en este formato, no podemos modificarla, sólo acceder a ella y leerla.

Claro, que siempre podremos regenerar el archivo creado con *squashfs* a partir de la información original, una vez modificada.

Este tipo de sistemas tiene grandes ventajas de cara a los sistemas live, ya que nos permite meter distribuciones completas en un CD normal. Y entre los sistemas existentes actualmente, éste es el que nos ha dado mejor rendimiento, mejor ratio de compresión (mayor cantidad de datos en el menos espacio posible) y más estable.

Pero no sólo sirve para sistemas live, aunque su uso en este tipo de sistemas se ha disparado en los últimos 2 años. De echo, se creo originariamente para sistemas muy pequeños empotrados como *Access Points* para redes *WiFi* (inalámbricas) y dispositivos similares.

### 4.2.2.- UNIONFS

Ya explicamos antes lo que es un *sistema de ficheros*, así que iremos directamente a explicar las peculiaridades de éste en concreto.

Si un sistema de ficheros estándar (*ext3*, *xfs*, *vfat*, *ntfs*, etc) interpreta y traduce los *clusters*, *sectores* y *bytes* en *ficheros* y *directorios*, el *unionfs* lo que hace es interpretar dos o más directorios, con su contenido, y

mostrarlos superpuestos como otro directorio, con su contenido.

Pero el contenido de este último directorio resultante es la diferencia entre los directorios que se está *interpretando*. Para verlo más claro, tenemos este gráfico:

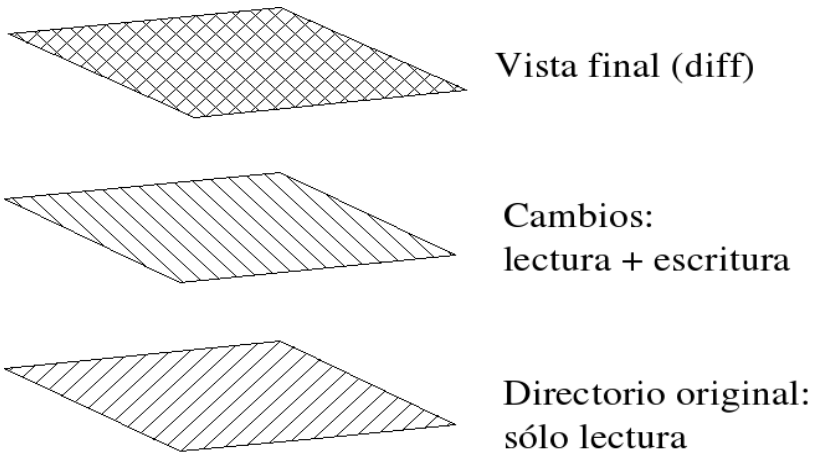



*Figura 13: Abstracción del funcionamiento del Unionfs y su vista resultante para el usuario y las aplicaciones.*

### 4.2.3.- SQUASHFS + UNIONFS

Hace ya algún tiempo que se viene usando el "squashfs" para los sistemas live, por razones obvias. Pero usar sistemas de ficheros y dispositivos de "sólo lectura" tiene bastantes limitaciones.

Por ejemplo, no podemos modificar configuraciones de programas o instalar nuevas aplicaciones. Eso sin contar con que preparar el sistema para funcionar en modo live se vuelve algo bastante complejo y lioso.

Con la suma de “Squashfs” y “Unionfs”, conseguimos meter una distribución entera en un CD y conseguir arrancarlos en modo live, de forma sencilla y pudiendo escribir sobre la distro (cambiar configuraciones a programas, instalar aplicaciones, etc). Esto aporte un enorme valor añadido a nuestra distribución y facilita mucho la tarea de mantenimiento y mejoras del propio sistema live.

## 4.3.- SISTEMA DE GENERACIÓN: GENLIVE, CHROOT, ETC

Para la generación de versiones *live* de la distribución se han creado sistema rudimentario pero efectivo, que nos automatizará casi todo el proceso. Esto no es necesario pero si altamente recomendable. Principalmente por dos razones:

- De esta manera nos resultará más complicado olvidarnos de partes del proceso.
- Si tenemos algún fallo, al ser un método estandarizado, será más rápidamente identificable por cualquier otra persona que nos pudiera ayudar.

Dentro del el sistema de generación hay dos partes importantes. Por un lado tenemos el software con el que se generan las imágenes y por otra dónde y como se prepara la distribución para poder crear éstas imágenes. Así que dividiremos y explicaremos el sistema en estas dos partes:

– Genlive

– Chroot

## 4.3.1.- GENLIVE

Es un pequeño y sencillo script que nos facilita la tarea de ejecutar ciertos comandos, para no tener que memorizar sus opciones y poder automatizar un poco más el proceso de generación de la imagen del CD *live*.

Es importante estar seguros de haber limpiado bien los restos que podamos haber dejado, de pruebas y demás, en los *sources* (*/media/distro/sources/* , lugar donde hemos instalado y configurado nuestra distro). Esto lo haremos con el script *clean.sh* que se proporciona (*/usr/share/genlive/clean.sh*) y cuyo uso explicaremos después.

El *script* se llama **genlive** y su uso es el que sigue:

```
$ genlive [opciones]
```

Como se puede observar, no es complicado, puesto que en la mayoría de los casos no es necesario pasarle ninguna opción al script.

Pero, ¿qué pasa cuando ejecutamos este *script*?

Pues, al margen de las comprobaciones básicas de si existen los archivos y directorios necesarios para crearse correctamente la distro, lo que se hace es crear el *initramfs* para configurado para arrancar en live, un archivo comprimido con el contenido de la distribución y generar la imagen del CD.

Por tanto las partes más importantes (bueno, las comprobaciones también lo son...) del *script* son:

Primero se copia el kernel y se genera el *initramfs* adecuado:

```
cp -a /boot/vmlinuz-${KERNEL} ${MASTER}/isolinux/vmlinuz
mkinitramfs -o ${MASTER}/isolinux/initramfs ${KERNEL}
```

Después se genera la imagen comprimida de la distro instalada en */media/distro/sources*:

```
mksquashfs ${SOURCES} ${MASTER}/META/META.squashfs
```

Y, por último, se genera la imagen del CD que nos permitirá arrancar en modo live:

```
mkisofs -l -r -J -V "${VOLUMENAME}" -hide-rr-moved -v -b isolinux/isolinux.bin -c boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o ${OUTPUTIMAGE} ${MASTER}
```

La definición de las variables usadas en las otras partes, se encuentran al principio del *script*:

```
KERNEL="2.6.12-9-386"
SOURCES="/media/distro/sources"
```

```
MASTER="/media/distro/master"
ISOS="/media/distro/isos"
IMAGEPREFIX="gl_v3"
OUTPUTIMAGE=${ISOS}/${IMAGEPREFIX}-$(date +%Y%m%d%H%M).iso
LOG_FILE="/var/log/installer.log"
SPLASH_IMAGE=""
VOLUMENAME="Guadalinex Live System"
```

Claro que estos valores pueden ser modificados directamente en el *script* (lo más cómodo y seguro si son cambios que se van a dar siempre) o pasándole ciertas opciones al **genlive**. Aquí se explican las opciones disponibles y cuales modifican esos datos:

- Opciones:
  - **-h** : ver las opciones.
- **-x** : no regenera la imagen comprimida de la distribución. Útil cuando ya hemos generado la imagen comprimida de la distribución y tenemos que generar otra iso, porque hemos cambiado cosas del *initramfs*, del *isolinux*, o de los *extras*, pero no de la distro. Esto nos ahorrará mucho tiempo en la regeneración del CD.
- **-y** : no regenera el initramfs. Útil cuando ya sabemos que ésta esta ya generada y no queremos perder tiempo con eso o arriesgarnos a que se genere mal.
- **-k <kernel version>** : especificar una versión de kernel. Se sustituirá la versión predeterminada (*2.6.12-9-386*) por la especificada aquí.
- **-s <splash image>** : especificar la imagen del arranque. Se copiará el archivo de imagen para *isolinux* especificado en el lugar correcto para poder ser usado.
- **-o <output image>** : especificar el nombre del archivo iso. Esto incluye su *path*, por supuesto. (Ej: */tmp/mi_iso.iso*)
- **-v <volume name>** : especificar el nombre del CD (es con el que se identifica el CD y que aparecerá en el *nautilus* al ser éste montado).
- **-p <image prefix>** : especificar el prefijo del nombre de la iso. El nombre por defecto es similar a "*gl_v3-200605081050.iso*" (cambiando la fecha y hora, naturalmente). Si especificáramos otro prefijo, como por

ejemplo "*test*", el resultado sería un nombre del archivo iso tal que: "*test-200605081050.iso*".

Las últimas cinco opciones pueden definirse de manera permanente dentro del script en las variables especificadas antes.

## 4.3.2.- PAQUETES Y SOFTWARE IMPLICADO

- **genlive**:
    - **clean.sh**: Éste es un script sencillo que se encarga de limpiar un poco el directorio donde tenemos instalada la distribución para que no de conflictos al arrancar o instalarse desde el *Live*. Está localizado en */usr/share/genlive/clean.sh*, instalado ahí por el paquete *genlive*, y es conveniente ejecutarlo antes de generar una imagen.
- **mksquashfs**: Este programa nos permite generar las imágenes comprimidas de las distribución instalada. Es usado por *genlive* y su modo de empleo fuera de este entorno muy sencillo es muy simple. Sólo hace falta indicarle dos parámetros: de qué directorio va a hacer la imagen y el nombre del archivo a generar:

```
$ mksquashfs /media/distro/sources /media/distro/master/META/META.squashfs
```

    - **mkisofs**: Por el nombre se puede deducir que no viene a ser muy diferente que el anterior, aunque para archivos iso, no para comprimir con *Squashfs*. Aquí las opciones son muchas y variadas, dependiendo de que tipo de CD y como es el contenido del mismo. Por eso lo mejor es dejar que *genlive* elija las opciones correctas por nosotros.
- **Syslinux** (-> **Isolinux**): El paquete *Syslinux* contiene el programa *Isolinux* y sus archivos de configuración. Es un gestor de arranque pensado para dispositivos extraíbles, preferentemente. Es decir, para disquetes, CDs, usbs, etc. Nosotros usaremos en concreto el *Isolinux* que está especialmente diseñado para CDs.
- **initramfs-tools** (guadalinex-live): Ésta es una versión realizada para *Guadalinex v3* del paquete oficial de *Ubuntu Breezy* con el mismo nombre. Se trata de un paquete que contiene lo necesario para generar los *initramfs* del sistema. La diferencia entre el nuestro y el original es que le hemos añadido los scripts y programas necesarios para que se pueda arrancar, también, en modo *live*. No nos extenderemos mucho más aquí ya que existe un subapartado entero para el initramfs.

- **busybox-initramfs** (guadalinex-live): Al igual que el paquete anterior, éste es una versión del original del *Ubuntu Breezy.* El paquete contiene el programa *busybox* compilado con las opciones necesarias para un *initramfs*. En nuestra versión, se han añadido algunas más, para poder usarlo en modo *live*.

El *busybox* es una *mini shell* que contiene dentro del mismo archivo ejecutable muchos programas. Es una manera de tener dentro de la shell los programas y no depender de programas externos.

Existen dos formas de usar los programas compilados dentro de esta *caja llena*. Una es llamando al *busybox*, seguido por el comando que queremos lanzar:

```
$ /usr/lib/initramfs-tools/bin/busybox ls
[listará el directorio como si hubiéramos ejecutado un ls normal]
```

Y la otra es crear un enlace al *busybox* con el nombre del comando que queremos usar:

```
% ln -s /usr/lib/initramfs-tools/bin/busybox /tmp/ls
% /tmp/ls
[listará el directorio como si hubiéramos ejecutado un ls normal]
```

## 4.3.3.- DIRECTORIOS Y ARCHIVOS IMPORTANTES

Ahora analizaremos los archivos y directorios principales del sistema de generación del CD *live*. Es bueno que estemos familiarizados con éstos, por si debemos modificar algo, bien en la personalización, bien porque queramos hacer mejoras.

### DIRECTORIOS

- **/media/distro/sources**

Es el directorio donde se guardará el contenido de nuestra distro. Es decir, nuestra distro, una vez ya instalada a partir de los paquetes y metapaquetes. Puede ser tanto un directorio que contenga la distro (porque la hemos copiado de algún sitio donde estaba instalada, o bien porque hemos instalado a partir de un *debootstrap*) como una partición, en donde hemos instalado nuestra distribución y que hemos montado en ese directorio.

- **/media/distro/master**

En este directorio se almacenará el contenido final del CD. Es lo que se usará como *master* a la hora de crear una imagen del CD. El directorio *master* contiene, a su vez, otros subdirectorios:

```
/media/distro/master/
                      |-- META/
                      |-- extras/
                      |    |-- disquete/
                      |    `-- info/
                      |         `-- documentos/
                      `-- isolinux/
```

*Figura 14: Árbol de directorios del sistema de generación de la versión Live*

- **META/**: Es el directorio donde se guarda la distribución comprimida en el archivo *META.squashfs*.
- **extras/**: Es un directorio opcional y prescindible, aunque útil e interesante, para poner información sobre la distro, documentación, archivos de ejemplo o lo que se quiera.
- **isolinux/**: Éste es el directorio en donde se guardará el *kernel*, el *initramfs* y todos los archivos de configuración del *isolinux*.
- **/media/distro/isos**

  Aquí se guardarán las imágenes de CDs (isos) generadas.

## ARCHIVOS

- **/media/distro/master/META/META.squashfs** = distro

Este archivo es una imagen comprimida de la distribución que previamente hemos instalado en */media/distro/sources*. El archivo contiene toda la distribución, con su jerarquía de directorios incluido, pero en un archivo de sólo lectura (sobre el que no se puede escribir), que está comprimido permitiendo, por ejemplo meter el contenido de una distribución que instalada ocupa 2 GB., en un archivo de poco más de 600 MB.

- **/media/distro/master/isolinux/initramfs** = initramfs

Como se ha explicado en capítulos anteriores, el *initramfs* es la pieza clave del sistema live y aquí es donde se guarda. Lo generará y guardará aquí el *genlive*.

- **/media/distro/master/isolinux/isolinux.bin** = sector de arranque

Es el sector de arranque del sistema *Isolinux*, que permite indicar a nuestro ordenador que podemos

arrancar desde el CD-ROM. Sin éste archivo y sin que se especifique al generar la iso, no se podría arrancar el CD.

- **/media/distro/master/isolinux/isolinux.cfg** = configuración

```
default linux
DISPLAY isolinux.txt

TIMEOUT 100
PROMPT 1

F1 isolinux.txt
F2 ayudaf2.txt
F3 ayudaf3.txt
F4 ayudaf4.txt
F5 ayudaf5.txt
F6 ayudaf6.txt
F7 ayudaf7.txt
F8 ayudaf8.txt
F9 ayudaf9.txt

KBDMAP es.kbd
FONT fuente.psf

label linux
  kernel vmlinuz
   append  ramdisk_size=100000  root=/dev/ram0  initrd=initramfs  BOOT=live
debug splash quiet
label db


  kernel vmlinuz
   append  ramdisk_size=100000  root=/dev/ram0  initrd=initramfs  BOOT=live
debug break
label single
  kernel vmlinuz
   append  ramdisk_size=100000  root=/dev/ram0  initrd=initramfs  BOOT=live
```

```
debug init=/bin/sh
      label failsafe
        kernel vmlinuz
          append ramdisk_size=100000 root=/dev/ram0 initrd=initramfs BOOT=live
debug noapic nolapic acpi=off pci=noacpi vga=normal xdriver=vesa
      label testcd
        kernel vmlinuz
          append ramdisk_size=100000 root=/dev/ram0 initrd=initramfs BOOT=live
debug md5 splash quiet
```

Éste es el archivo de configuración de *isolinux* para el CD *live*. Las opciones generales del *isolinux* ya han sido vistas anteriormente en el apartado sobre la modificación de la distribución, así que ahora nos centraremos exclusivamente en los detalles que afecten al sistema *live*.

Pero dentro de los parámetros que se le pasan al kernel (con el *append*) hay algunos que sólo existen en nuestro sistema live y otros que son parámetros normales, aunque necesarios para que esto funcione. Vamos a explicarlos por separado, empezando por estos últimos:

- ramdisk_size=100000: Este parámetro especifica el tamaño, en KB., del sistema inicial que se monta en la memoria ram. Generalmente necesitamos especificar uno mayor que el predeterminado (4096 KB. = 4 MB.), por eso lo especificaremos.
- **root=/dev/ram0**: El *root* indica la partición física en donde se montará el sistema raíz (*/*), que para nuestro sistema live será la memoria ram, por eso necesitaremos indicar */dev/ram0*.
- **initrd=**: Aquí se especifica el archivo *initrd* o *initramfs* que se usará. En nuestro caso es el *initramfs* generado y con todo lo necesario arpa montar el sistema live.
- **quiet**: Si se especifica dicho parámetro, no se mostrarán todos los mensajes del kernel al arrancar, sólo los errores y mensajes más importantes.
- **debug**: Este parámetro lo usan los scripts del *initramfs* para ver si queremos ir viendo y guardando información de lo que se va ejecutando y los errores que se den. Es una opción para depurar errores. Si existe el parámetro se activará este modo de depuración, en caso contrario no se mostrará esa información. En caso de que se parara la ejecución en el *initramfs*, podremos ver esta información en

*/tmp/initramfs.debug*.

- **splash**: Si existe este parámetro en la línea del *append*, se mostrará el arranque gráfico, si no fuese así, se mostraría el arranque normal, en modo texto.
- **break**: Dentro del *initramfs*, hay un punto intermedio en la ejecución en que se comprueba si se ha pasado esta opción. De ser así, se detendrá la ejecución en ese punto, mostrando una consola con comandos mínimos, para poder hacer pruebas y depurar. Es útil para la depuración de ciertas partes.

Y aquí están los exclusivos del sistema live:

- **BOOT=live**: El sistema de scripts del *initramfs* soporta varios tipos de arranque (*local y nfs*), que se indican con esta variable que se pasa al kernel, pero nosotros hemos añadido un nuevo tipo de arranque, que es el arranque *live* y que se especifica como ahí se indica (*BOOT=live*).
- **xdriver=**: Con esta variable se puede especificar un driver determinado para las X. Esto puede ser útil en equipos con dificultades para detectar dicho driver o en los que la tarjeta gráfica no está bien soportada. En este último caso, se le puede indicar el driver genérico *vesa*, que no optimizará su rendimiento, pero asegurará que arranque.
- **md5**: Ésta es una opción que le indica al *initramfs* que debe hacer una comprobación de la integridad de los archivos contenidos en el CD. Para ello se comprueba una lista de *sumas md5* (*checksum*) de los archivos del CD, que viene incluido en el mismo.
- **/media/distro/master/isolinux/splash.rle** = imagen del arranque

Este archivo se corresponde con la imagen inicial que aparece cuando se carga el CD. Para ver como se modificaría esta imagen véase la sección sobre el *Isolinux* en el capítulo de *personalización*.

- **/media/distro/master/isolinux/ayuda#.txt** = archivos de ayuda

Existen varios archivos de ayuda, cada uno está asignado (como vimos en el archivo de configuración *isolinux.cfg*) a una *tecla de función* (F2, F3, F4, etc). Se pueden reconocer fácilmente por el nombre y por estar especificados en el citado archivo de configuración.

Estos archivos, al igual que el presentación (*isolinux.txt*) pueden tener, y de hecho tienen, algunos códigos especiales para formatear el texto. En los archivos de ejemplo aquí mostrados se puede ver claramente. Son códigos del tipo: 09, 07, 0d...

Véanse aquí los archivos de ayuda de *Guadalinex v3*, a modo de ejemplo:

**isolinux.txt**

```
^Xsplash.rle
```

```
^O07Bienvenido a Guadalinex v3 (live), pulse INTRO para continuar.
Para ver la ayuda, pulse F2.
```

**ayudaf2.txt**

```
                              09Índice de ayuda07


Dispone de varias páginas de ayuda:

     0dF107 ............. 0dPantalla inicial07
     0dF207 ............. 0dÍndice de ayuda07
     0dF307 ............. 0d¿Qué es Guadalinex?07
     0dF407 ............. 0dIniciando Guadalinex v307
     0dF507 ............. 0dConfiguración adicional07
     0dF607 ............. 0dParámetros07
     0dF707 ............. 0dOpciones de arranque07
     0dF907 ............. 0dGuadalinex en Internet07


Presione 0dINTRO07 para iniciar Guadalinex v307
```

**ayudaf3.txt**

09¿Qué es Guadalinex?07

Bienvenido/a al sistema operativo GuadaLinex, una distribución de 0dGNU/Linux
07elaborada por la 0dJunta de Andalucía07.

Puede utilizar esta distribución cuantas veces quiera en modo 0d"Live"07, lo que significa que no se modificará nada en su disco duro.
Cuando reinicie el ordenador y extraiga el CD-ROM de la unidad, podrá acceder
a su sistema operativo de siempre, sin rastro alguno de GuadaLinex.

En todo momento dispone de la posibilidad de instalar este sistema operativo en su disco duro, conviviendo con el resto de sistemas operativos que tenga instalados. Para ello, sólo tiene que pulsar el botón 0d"Instalador de GuadaLinex"07 en el escritorio.

07Pulse 0dF207 para volver al Índice de ayuda
Presione 0dINTRO07 para iniciar Guadalinex v307

**ayudaf4.txt**

09Iniciando Guadalinex v307

Iniciar Guadalinex v3 en modo 0d"live"07 (desde el CD) es muy sencillo, únicamente debe insertar el CD-ROM en el ordenador y reiniciar.
Se le mostrará una pantalla de presentación y en unos segundos se cargará el sistema operativo.

Si Guadalinex no se iniciase correctamente, es posible que deba modificar el 0dorden de arranque07 de los dispositivos en la 0dBIOS07 de su ordenador.
Este procedimiento varía ligeramente entre distintos modelos así que se recomienda consultar en el manual del ordenador la forma correcta de hacerlo.

07Pulse 0dF207 para volver al Índice de ayuda
Presione 0dINTRO07 para iniciar Guadalinex v307

**ayudaf5.txt**

09Configuración adicional07

Guadalinex v3 está preparado para detectar de forma 0dautomática07 el hardware de su ordenador pero, en algunas configuraciones, es posible que el arranque falle. En estos casos, es necesario indicarle al sistema operativo algunos datos sobre nuestro ordenador para facilitar el arranque.

Para pasarle esta información a Guadalinex se debe utilizar el indicador 0d"boot:"07 situado en la parte inferior de la pantalla. En este indicador debemos escribir 0d"linux"07 seguido de los parámetros deseados.

Puede consultar los posibles valores en la sección Parámetros (tecla 0dF607).

07Pulse 0dF207 para volver al Índice de ayuda
Presione 0dINTRO07 para iniciar Guadalinex v307

**ayudaf6.txt**

```
                                    09Parámetros07



      0dacpi=off07: Deshabilita el soporte para control de energía por ACPI
                  Ej: linux acpi=off
      0dnoapic07:   Deshabilitado el módulo de temporización APIC
                  Ej: linux noapic
      0dnolapic07:  Pruebe este opción si la anterior falla
                  Ej: linux nolapic
      0dnousb07:    Deshabilita el subsistema USB
                  Ej: linux nousb
      0dxdriver=07: Establece el driver para el sistema X Window
                  Ej: linux xdriver=vesa


                                                        (Más ayuda... F7)


      Se pueden 0dcombinar07 varios parámetros separándolos por espacios, por
ejemplo:
      linux acpi=off noapic nolapic nousb




      07Pulse 0dF207 para volver al Índice de ayuda
      Presione 0dINTRO07 para iniciar Guadalinex v307
```

**ayudaf7.txt**

09Opciones de arranque07

0dfailsafe07: Esto no es un parámetro, sino una opción completa preconfigurada para ordenadores con muchos problemas (no carga usplash entre otras cosas).
Si tiene problemas con el arranque, utilice esta opción directamente.

0dsingle07: Con esta opción, Guadalinex arranca en modo monousuario sin ejecutar muchos de los programas que carga habitualmente (incluyendo el entorno gráfico). Utilice esta opción si tiene problemas cargando el modo gráfico.

0ddb07: Esta es una opción para usuarios experimentados y desarrolladores que se utiliza para iniciar el sistema hasta la primera etapa del proceso de arranque.

0dtestcd07: Con esta opción se comprueba la integridad del CD, para ver que no esté mal grabado o en mal estado.

07Pulse 0dF207 para volver al Índice de ayuda
Presione 0dINTRO07 para iniciar Guadalinex v307

**ayudaf9.txt**

09Guadalinex en Internet07

El proyecto Guadalinex dispone de un sitio 0dweb07 en Internet donde podrá descargar las 0dnuevas versiones07, consultar documentación, participar en foros y listas de correo, etc.

La dirección de Guadalinex es:
0dhttp://www.guadalinex.org07

07Pulse 0dF207 para volver al Índice de ayuda
Presione 0dINTRO07 para iniciar Guadalinex v307

## 4.3.4.- CHROOT

Una vez tenemos nuestra distribución debemos instalarla para probarla. A partir de aquí entraremos en una fase de muchas pruebas y de configuraciones que nos ayudarán a crear los paquetes definitivos y a partir de estos, las versiones finales de la distribución. En sus dos facetas: "live" y "no live".

Hay diversas maneras de hacer esto, pero la que recomendamos es instalar, a partir del instalador "no live" en una partición del disco que usaremos para generar nuestras distros.

La idea es que podamos montar esa partición, ya instalada, en nuestro sistema de trabajo y así poder hacer pruebas, cambios y generar a partir de ese directorio ("/media/distro/sources") nuestros CDs "live".

Y la forma más fácil y cómoda para hacer pruebas, instalar paquetes, cambiar archivos de configuración, etc, en esa partición sin tener que reiniciar todo el rato es usar el programa "chroot". El cual sirve para poder ejecutar otras aplicaciones dentro de un directorio indicado como si éste fuera el directorio raíz del sistema.

Es decir, que si ejecutáramos un programa, por ejemplo, una "shell" (el programa predeterminado si no se indica otro) estaremos en una "shell" dentro de ese directorio, sin posibilidad de subir niveles u obtener información sobre nada que esté por encima de nuestro, ahora, directorio *raíz*.

Dicho programa es muy útil para poder instalar paquetes en un sistema instalado en un directorio, como si se hubiera hecho arrancando esa partición.

La forma de usar ese programa es:

```
$ chroot /path/del/directorio/ [comando]
```

El *comando* podemos omitirlo si lo que queremos es una "shell" dentro del sistema. Eso sí, deberemos ser *root* para poder usar este programa.

Pero no todo es bonito y maravilloso con el "chroot". Nos ira muy bien para instalar paquetes, hacer cambios en archivos de configuración y demás, pero podría darnos problemas si jugamos con las *X* o estamos arrancando y parando servicios. también si jugamos con el kernel, que por otro lado debe ser el mismo en ambos sistemas. No podemos arrancar una distro dentro de otra ya arrancada, con kernels diferentes.

Estas limitaciones se deben a que ambos sistemas, el "anfitrión" y el "huésped", comparten el mismo kernel, variables de entorno e información del "proc". Así que si usamos o tocamos dicha información en uno,

podremos estar haciéndolo en el otro.

También deberemos tener algunas cosas en cuenta a la hora de usar el "chroot" para entrar en un sistema como si lo arrancáramos y queramos hacer cosas dentro de él. Como por ejemplo, debemos acordarnos de montar el *sistema de ficheros virtual* "proc" en el directorio "/proc".

El "proc" es la interfaz de comunicación entre el kernel, el hardware y los programas que se ejecutan en el sistema. En el podemos ver, a modo de ficheros y directorios, información sobre los distintos dispositivos conectados a nuestro equipo y de los que el kernel tiene conocimiento. También podemos ver los "procesos" que se están ejecutando en nuestro sistema (existe un directorio, con su correspondiente número, por cada proceso), así como información del tipo, propietario del proceso, tiempo de ejecución, recursos que está usando, otros procesos de los que depende, etc.

Y esta *interfaz* es usada por muchos programas para obtener información del sistema. Éste es el caso de programas tan básicos y necesarios como "mount", "apt", etc.

Pero no debemos olvidarnos de desmontar dicho directorio cuando acabemos. De no hacerlo así, no podríamos desmontar la partición que usamos para nuestra distro, o podríamos copiar un montón de archivos en la imagen del sistema "live", que no harían más que ocupar bastante espacio y podrían causar problemas en la distro arrancada.

Así que la secuencia lógica para usar el "chroot" en nuestro propósitos sería la siguiente:

```
$ chroot /media/distro/sources/
[ya estamos dentro]
chroot$ mount -t proc none /proc
[instalamos y modificamos a gusto]
chroot$ apt-get install [algo]
...
chroot$ umount /proc
chroot$ exit
[ya estamos fuera]
```

Otro detalle a tener en cuenta es que probablemente en el sistema que tengamos instalado en ese directorio ("/media/distro/sources"), no tengamos configurados los servidores de nombres ("DNS"), así que si intentamos usar el "apt-get", nos podría dar un error. Eso se arregla fácilmente copiando el contenido de nuestro archivo "/etc/resolv.conf" dentro del de la distro instalada "/media/distro/sources/etc/resolv.conf".

## 4.4.- ¿DÓNDE PERSONALIZAR?

Ya hemos estado viendo algunos sitios donde se puede modificar el sistema live, pero vamos a mencionar alguno más en esta sección. Éstos serían cosas bastante específicas del sistema live como las siguientes:

- **Usplash**: En el capítulo dedicado a los aspectos modificables de la distro, se ha comentado ya como se cambia o personaliza el "usplash", pero lo que queremos recalcar aquí es que es **en el sistema en donde se genera la distribución**, donde tiene que estar instalada esa imagen a usar para el "usplash".

  Es decir, si nosotros tenemos un ordenador con un sistema linux instalado, en el que hemos instalado el "genlive" y demás paquetes para generar una distribución, es en ese sistema en donde tendremos que instalar ese paquete recompilado del "usplash".

  Por supuesto, la distribución debe llevar el paquete ya modificado, también.

- **[initramfs]/scripts/live-bottom/hacks**: Este script se encuentra dentro del "initramfs" y su función es facilitarnos el añadir cambios de prueba o no estándares, para el arranque del sistema live.

  Algunos ejemplos de esto pueden ser, añadir un icono al escritorio del usuario que se arranca, lanzar un servicio ("SAMBA", "apache", etc) antes de abrir la sesión del usuario. Cambiar el escritorio con que se inicia la sesión del usuario, etc.

- **/etc/mkinitramfs/initramfs.conf**: En este archivo existe una variable que se puede configurar, que es el nombre que tendrá el usuario al arrancar el sistema live. La variable es "USERNAME=".

# 4.5.- INSTALADOR LIVE

## 4.5.1.- ¿QUÉ ES?

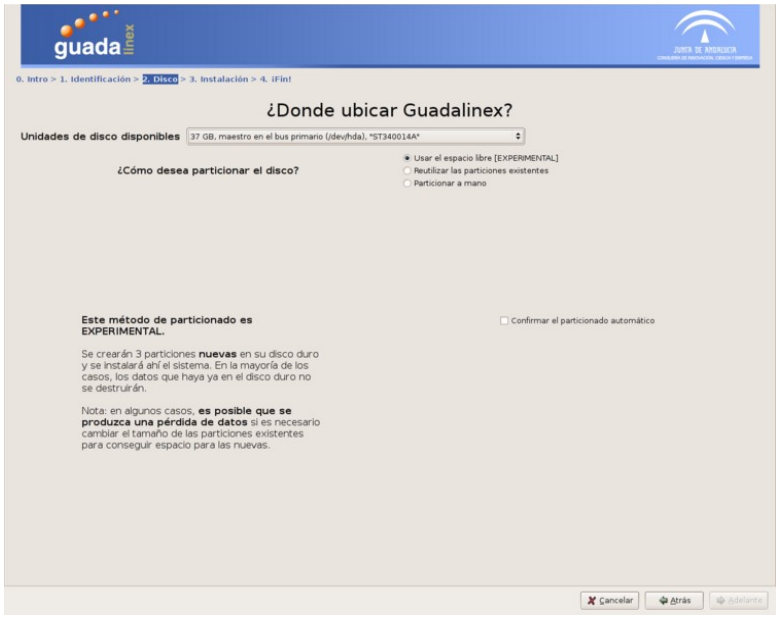
Lo bueno de un sistema live es que se puede probar antes de instalar, pero también que se puede instalar de manera más rápida y sencilla para el usuario. Hay muchos instaladores para sistemas live, aunque casi todos tienen una limitación principal, que se limitan a copiar la distribución y configuraciones. Es decir, que no usan el sistema de paquetería o herramientas propias de la distribución para configurar programas y hardware.

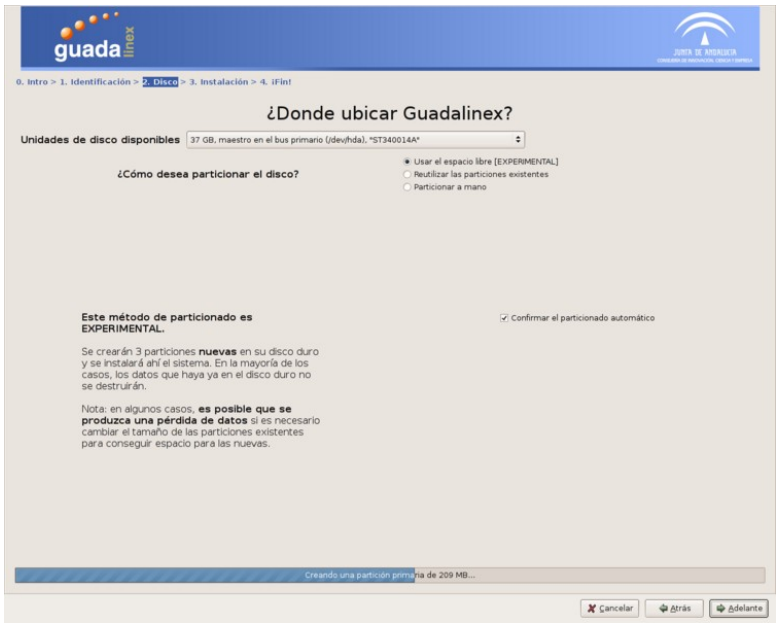
Esto puede generar muchos problemas, como por ejemplo en las actualizaciones. Si hay algo configurado por un script propio, puede que el sistema no sea consciente de ellos y lo machaque al actualizar el paquete al que pertenece.

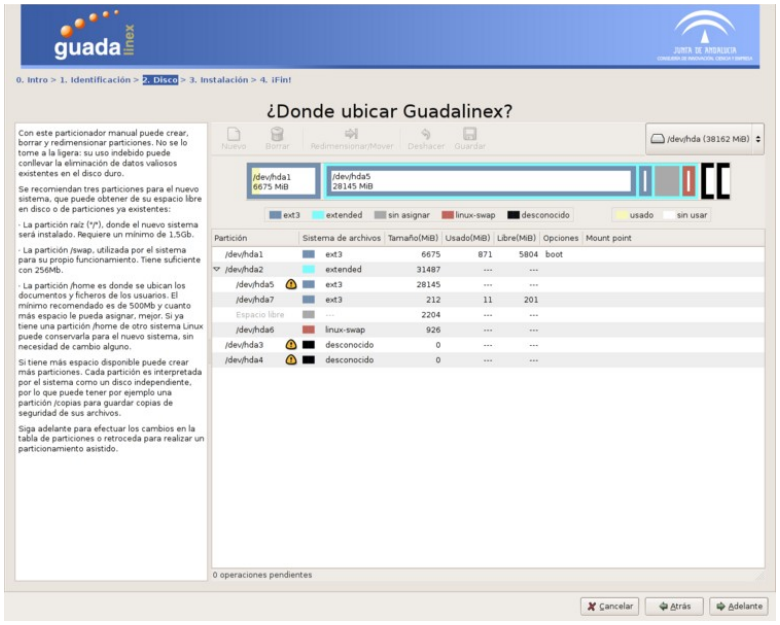
Por esta razón hemos diseñado un instalador que use en, la medida de lo posible, las herramientas de la distribución base, para configurar la distro copiada al disco. Además de crear un sistema de particionado bastante inteligente que facilita una de las tareas que más miedo produce a un usuario a la hora de instalar. Y aprovechando que disponemos de una distribución completa arrancada, hemos creado una interfaz de usuario agradable y sencilla, para facilitarle la vida al usuario. Eliminando todas las preguntas que no sean absolutamente necesarias.

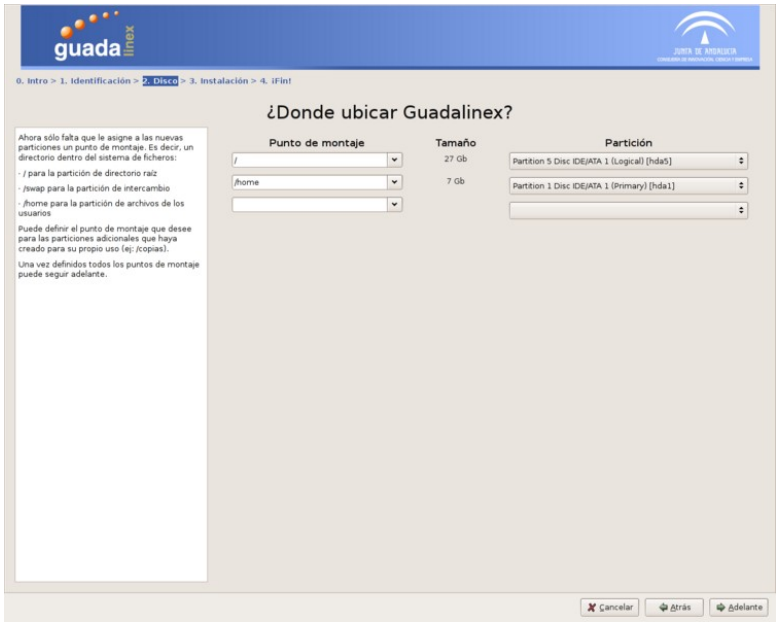

guadalinex
0. Intro > 1. Identificación > 2. Disco > 3. Instalación > 4. ¡Fin!
Guadalinex Express
La manera más rápida de instalar Guadalinex en su ordenador
Se dispone a instalar el sistema Guadalinex que está probando. Es un proceso sencillo de 3 pasos, se realiza en unos 15 minutos y precisa un mínimo de 1,5Gb de espacio libre en su disco, si bien es recomendable llegar a 5GB si va a instalar muchos programas adicionales.
Consulte la guía rápida para ver cómo se desarrolla la instalación.
Cancelar
Adelante


guadalinex
0. Intro > 1. Identificación > 2. Disco > 3. Instalación > 4. ¡Fin!
Identifíquese y ponga nombre a su ordenador
Datos personales
Nombre completo
Usuario
Contraseña
Verifique contraseña
Datos del ordenador
Nombre
Es necesario que introduzca su nombre de usuario para el sistema, su nombre completo para generar una ficha de usuario, así como el nombre de máquina con el que quiera bautizar su equipo. Deberá teclear la contraseña de usuario en dos ocasiones.
Cancelar
Adelante


guadalinex
0. Intro > 1. Identificación > 2. Disco > 3. Instalación > 4. ¡Fin!
¿Donde ubicar Guadalinex?
Unidades de disco disponibles
37 GB, maestro en el bus primario (/dev/hda), *ST340014A*
¿Cómo desea particionar el disco?
Usar el espacio libre [EXPERIMENTAL]
Reutilizar las particiones existentes
Particionar a mano
Este método de particionado es EXPERIMENTAL.
Se crearán 3 particiones nuevas en su disco duro y se instalará ahí el sistema. En la mayoría de los casos, los datos que haya ya en el disco duro no se destruirán.
Nota: en algunos casos, es posible que se produzca una pérdida de datos si es necesario cambiar el tamaño de las particiones existentes para conseguir espacio para las nuevas.
Confirmar el particionado automático
Cancelar
Atrás
Adelante

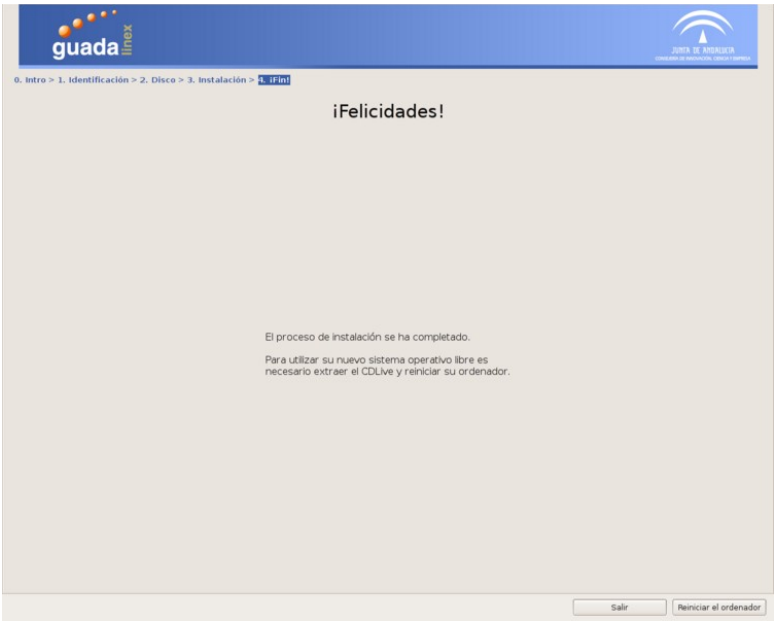

guadalinex
0. Intro > 1. Identificación > 2. Disco > 3. Instalación > 4. ¡Fin!
¿Donde ubicar Guadalinex?
Unidades de disco disponibles
37 GB, maestro en el bus primario (/dev/hda), *ST340014A*
¿Cómo desea particionar el disco?
Usar el espacio libre [EXPERIMENTAL]
Reutilizar las particiones existentes
Particionar a mano
Este método de particionado es EXPERIMENTAL.
Se crearán 3 particiones nuevas en su disco duro y se instalará ahí el sistema. En la mayoría de los casos, los datos que haya ya en el disco duro no se destruirán.
Nota: en algunos casos, es posible que se produzca una pérdida de datos si es necesario cambiar el tamaño de las particiones existentes para conseguir espacio para las nuevas.
Confirmar el particionado automático
Creando una partición primaria de 209 MB...
Cancelar
Atrás
Adelante


guadalinex
0. Intro > 1. Identificación > 2. Disco > 3. Instalación > 4. ¡Fin!
¿Donde ubicar Guadalinex?
Con este particionador manual puede crear, borrar y redimensionar particiones. No se lo tome a la ligera: su uso indebido puede conllevar la eliminación de datos valiosos existentes en el disco duro.
Se recomiendan tres particiones para el nuevo sistema, que puede obtener de su espacio libre en disco o de particiones ya existentes:
· La partición raíz ("/"), donde el nuevo sistema será instalado. Requiere un mínimo de 1,5Gb.
· La partición /swap, utilizada por el sistema para su propio funcionamiento. Tiene suficiente con 256Mb.
· La partición /home es donde se ubican los documentos y ficheros de los usuarios. El mínimo recomendado es de 500Mb y cuanto más espacio le pueda asignar, mejor. Si ya tiene una partición /home de otro sistema Linux puede conservarla para el nuevo sistema, sin necesidad de cambio alguno.
Si tiene más espacio disponible puede crear más particiones. Cada partición es interpretada por el sistema como un disco independiente, por lo que puede tener por ejemplo una partición /copias para guardar copias de seguridad de sus archivos.
Siga adelante para efectuar los cambios en la tabla de particiones o retroceda para realizar un particionamiento asistido.
/dev/hda (38162 MiB)
Nuevo
Borrar
Redimensionar/Mover
Deshacer
Guardar
/dev/hda1
6675 MiB
/dev/hda5
28145 MiB
ext3
extended
sin asignar
linux-swap
desconocido
usado
sin usar
Partición
Sistema de archivos
Tamaño(MiB)
Usado(MiB)
Libre(MiB)
Opciones
Mount point
/dev/hda1 ext3 6675 871 5804 boot
/dev/hda2 extended 31487 --- ---
/dev/hda5 ext3 28145 --- ---
/dev/hda7 ext3 212 11 201
Espacio libre 2204 --- ---
/dev/hda6 linux-swap 926 --- ---
/dev/hda3 desconocido 0 --- ---
/dev/hda4 desconocido 0 --- ---
0 operaciones pendientes
Cancelar
Atrás
Adelante


guadalinex
0. Intro > 1. Identificación > 2. Disco > 3. Instalación > 4. ¡Fin!
¿Donde ubicar Guadalinex?
Ahora sólo falta que le asigne a las nuevas particiones un punto de montaje. Es decir, un directorio dentro del sistema de ficheros:
· / para la partición de directorio raíz
· /swap para la partición de intercambio
· /home para la partición de archivos de los usuarios
Puede definir el punto de montaje que desee para las particiones adicionales que haya creado para su propio uso (ej: /copias).
Una vez definidos todos los puntos de montaje puede seguir adelante.
Punto de montaje
Tamaño
Partición
/
27 Gb
Partition 5 Disc IDE/ATA 1 (Logical) [hda5]
/home
7 Gb
Partition 1 Disc IDE/ATA 1 (Primary) [hda1]
Cancelar
Atrás
Adelante


guadalinex
0. Intro > 1. Identificación > 2. Disco > 3. Instalación > 4. ¡Fin!
Cómo será su nuevo escritorio libre
Puede bajar fotos de su cámara digital, grabar música en su reproductor mp3 o gestionar su memoria USB simplemente conectando los dispositivos a su ordenador.
Preparando el directorio de instalación
Cancelar
Adelante


guadalinex
0. Intro > 1. Identificación > 2. Disco > 3. Instalación > 4. ¡Fin!
Cómo será su nuevo escritorio libre
Importe las canciones de sus CD, organícelas por autores, géneros, fechas o preferencias personales. También puede crear su propia música y compartirla.
No reproduciendo
Copiando 41% - Queda 03:23 - [usr/X11R6/lib/modules/drivers/trident_drv.o]
Cancelar
Adelante




## 4.5.2.- ¿CÓMO FUNCIONA?

El instalador de Guadalinex v3 está diseñado de tal forma que la lógica del instalador("backend") está separada de la parte que interactúa con el usuario para pedirle datos("frontend"). De esta forma, se podría fácilmente crear diferentes interfaces de interacción con el usuario.

Para nuestro caso hemos elegido una interfaz gráfica usando las librerías Gtk, ya que el entorno Guadalinex es Gnome y Gtk. Y tanto el "backend" como el "frontend" están programados en Python, para facilitar su desarrollo, mantenimiento y por su flexibilidad.

La parte que más nos interesará, sin duda, será el "frontend", que es la que podemos personalizar y que es la que ve el usuario. Dicha interfaz gráfica esta diseñada con "Glade", un programa para diseñar interfaces gráficas para Gnome/Gtk. Con el no sería complicado modificar la apariencia del instalador, aunque si no tenemos cuidado, puede darnos más problemas que ayudas.

Lo ideal es usar los elementos preparados para ser personalizados.

## 4.5.3.- ¿CÓMO PERSONALIZARLO?

Como mencionamos antes, hay ciertos elementos preparados para poder ser personalizados, como por ejemplo, la imagen de la cabecera, o el texto de bienvenida. Vamos a enumerarlos e indicar donde podemos encontrarlos:

- **Nombre de la distro:** Esto debe estar configurado en el "/etc/lsb-release" por el correspondiente paquete, ya que el instalador, buscará de aquí el nombre de la distribución. En concreto buscará el "DISTRIB_ID", pero poniéndolo todo en minúsculas. Con este nombre existirán varios directorios dentro del directorio fuente del paquete que se describen a continuación.

- ***glade/locale/[nombre_distro]***: Contiene las traducciones y ayudas de los textos mostrados en la interfaz gráfica.

- ***glade/pixmaps/[nombre_distro]***: Contiene las imágenes de la cabecera, y las capturas de aplicaciones que se muestran en el proceso de copia al disco. Las primeras querremos cambiarlas seguramente, las segundas dependerá del caso.

  Quizás queramos añadir alguna aplicación, quitar otra, o actualizar ejemplos. En cualquier caso debemos que tener en cuenta el archivo "messages.txt", en el cual se describe cada una de esas capturas.

  Debe ponerse un párrafo por captura de aplicación y hay que tener en cuenta que irán apareciendo en el orden puesto en el archivo, luego habrá que coordinar el número de línea con el número de captura (snapshot1.png, snapshot2.png, snapshot3.png, etc).

- ***htmldocs/[nombre_distro]***: Aquí se pondrá la presentación en "html" que aparece al principio.

Con estos elementos personalizados, le habremos cambiado la cara al instalador, dándole un aspecto profesional a nuestra distribución, que estará lista para ser probada e instalada.

# 5.- CONCLUSIONES

El equipo de desarrollo de Guadalinex v3 y los docentes del curso les animamos a que prueban los diversos sistemas que aquí se muestran. Sabemos que el conocimiento es extenso y es complicado asimilarlo en un curso tan reducido, pero estamos abiertos a cualquier duda mediante las listas oficiales[1] del proyecto[2].

Asimismo recomendamos que cualquier proyecto que se haga basado en éste, se intente desarrollar en este entorno (la forja de Guadalinex) o que se mantenga informado al equipo de desarrollo y a la comunidad mediante las listas.

Esto es importante para poder reaprovechar el trabajo de todos en beneficio de todos y para evitar que se tengan que pelear con cosas que pueden estar ya resueltas por otra persona.

En cualquier caso, quedamos a su disposición para cualquier duda, consulta o sugerencia.

1 http://forja.guadalinex.org/mailman/listinfo/forja.guadalinex2005-distro

2 http://forja.guadalinex.org/repositorio/projects/guadalinex2005/