# Construcción de un sistema Linux a partir de código fuente

Borja Requejo García
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Memoria del Proyecto Final de Carrera
Enginyeria Tècnica en Informàtica de Sistemes
Enero 2012

- **Datos del proyecto**

  Título: Construcción de un sistema Linux a partir de código fuente

  Estudiante: Borja Requejo García

  Titulación: Enginyeria Tècnica en Informàtica de Sistemes

  Créditos: 22,5

  Departamento: AC

- **Tribunal que evalúa este proyecto**

  Presidente: David López Álvarez

  Vocal: María Teresa Abad Soriano

  Director: Xavier Martorell Bofill

Fecha de lectura: 24 de enero de 2012

# Índice

# 1. Introducción

La informática ha avanzado mucho desde el momento en que apareció y nos ha proporcionado facilidades para hacer muchísimas tareas, pero sigue habiendo problemas que surgen de imprevisto, los cuales no son fáciles de detectar, prevenir, o arreglar sin los mecanismos adecuados.

Siempre es útil disponer de un sistema listo para arrancar de forma sencilla en cualquiera de las máquinas de que disponga nuestra organización, para examinar la situación de la propia máquina o de la red a la que está conectada.

En este proyecto, queremos proporcionar dicho sistema, construyéndolo desde código fuente, para determinar qué software es necesario, de dónde se puede obtener y cómo hay que configurarlo, compilarlo y ejecutarlo.

# 2. Objetivos

Para este proyecto nos hemos planteado conseguir los siguientes objetivos:

- Seleccionar las herramientas necesarias para construir un sistema completo a partir de código fuente, incluyendo el *kernel* del Linux y las utilidades de compilación de GNU.
- Instalar un compilador de GNU (GCC) reciente, a partir del código fuente, y eligiendo las características necesarias en cuanto a generación de código.
- Estudiar las opciones de configuración del *kernel* de Linux.
- Configurar y compilar el *kernel* de Linux, con únicamente las opciones necesarias para el sistema que se cree.
- Instalar un compilador para el sistema utilizando el nuevo compilador. Por lo tanto, todas las librerías que utilice deben haber sido compilados también por el nuevo compilador.
- Instalar los paquetes necesarios con el compilador del sistema para que el sistema funcione.
- Generar un sistema de ficheros apropiado con todas las utilidades, sistema operativo y gestor de *boot*.

Se propone realizar 2 versiones del sistema, con utilidades diferentes:

Versión 1: Sistema simple que ocupe poco espacio y que servirá para poder arreglar y buscar errores en otro sistema:

- Instalar un mínimo de paquetes complementarios útiles para el sistema.
- Conseguir que el sistema arranque desde un dispositivo USB
    - sin necesidad de disponer de distintos *run-levels* en el sistema operativo.

Versión 2: Crear un sistema básico que se pueda aumentar como si se tratase de una distribución de Linux normal (Ubuntu, Fedora...):

- Instalar paquetes necesarios para el arranque y montaje del sistema.

- Conseguir que el sistema arranque desde un dispositivo USB (con *run-levels*).
- Instalar paquetes complementarios.
- Conectarlo a Internet.
- Conseguir que sea un sistema autosuficiente y ampliable.

# 3. Análisis de tecnologías requeridas

Para llevar a cabo el proyecto hace falta:

- Una máquina con un SO Linux con potencia suficiente para soportar la fase de compilación del sistema y las herramientas a instalar.
- Un dispositivo de almacenamiento externo con un mínimo de 600 MiB para la versión 1, y de 1.2 MiB para la versión 2. Preferiblemente, que sea portable (pen drive...), ya que su función principal será para el tratamiento de problemas y emergencias con las máquinas en un entorno cambiante.
- Conexión a internet para la descarga de los paquetes a instalar en el sistema y realizar las pruebas del sistema de análisis.
- Código fuente del sistema operativo.
- Compilador y herramientas de soporte a la compilación.
- Código fuente de las herramientas del sistema y programas a instalar en el sistema.

# 4. Tecnologías utilizadas

Teniendo en cuenta las tecnologías requeridas, para el proyecto se ha utilizado:

- Hardware
  - Ordenador portátil Toshiba Satellite U500 con SO Ubuntu11.04 x86.
    - Procesador: Intel Core i5 (2,26 Ghz, 3Mb. de cache).
    - Memoria RAM: 4 GiB DDR3 (1.066 Mhz).
    - Disco duro: 500 GiB (5400 rpm).
  - Máquina virtual con el mismo SO para más facilidad a la hora de tratar problemas con el montaje y backups.
  - Conexión ADSL de 3 Mbit.
  - Pen drive de 4 GB.

- Software
  - Linux Kernel version 2.6.38-8
  - Todos los paquetes de software útiles para el proyecto que se han podido obtener de forma libre de la GNU Software Foundation (ver la lista en las secciones 7.2.1, 7.2.2 -> Programas necesarios y 7.2.3 -> Programas necesarios).
  - Otros paquetes de software útiles para el proyecto obtenidos de otras fuentes (ver la lista en las secciones 7.2.1, 7.2.2 -> Programas necesarios y 7.2.3 -> Programas necesarios).
  - Las aplicaciones necesarias para realizar el análisis del estado de otros ordenadores y de la red (ver la lista en la sección 7.2.3 -> Programas necesarios).

# 5. Especificación

Antes de comenzar a explicar qué componentes lleva cada versión, hay que aclarar que la versión 2 es una ampliación de la versión 1, todos los paquetes de la versión 1 están contenidos en la versión 2.

Para conseguir un sistema funcional, debemos incluir los siguientes componentes, los cuales explicaré individualmente más adelante.

## 5.1. Versión 1

Entorno de compilación

- binutils
- gcc
- glibc

Intérprete de comandos

- readline
- bash

Herramientas y comandos básicos del sistema:
En este grupo se incluyen comandos que están presentes en todos los sistemas Linux y que los ofrecen el paquete coreutils(cd, cp, mkdir, ls, ln, df...).

Edición de ficheros de texto

- vim

Herramientas de búsqueda de ficheros

- findutils

Procesado de texto

- grep
- sed
- gawk
- less

Herramientas de disco

- util-linux (fdisk, mount...)
- e2fsprogs (fsck, mkfs)

Herramientas de red

- net-tools (ifconfig, route, arp...)
- ping
- tcpdump

## 5.2. Versión 2

La versión 2 incluye todas las herramientas anteriores, y además:

X-Window

- xfree86
- fluxbox

Scripting

- perl
- PHP

Soporte para procesado de XML

- libxml2

Navegación web

- dillo

Servidor de Web

- apache

Monitor de servicios y recursos

- nagios-core
- nagios-plugins

Run-levels

- sysVinit (init, shutdown, halt, sulogin, last...)
- LFS-Bootscripts

Herramientas de sistema

- udev
- sysklogd
- module-init-tools (insmod, modprobe...)
- grub
- procps (ps, top...)

Herramientas de empaquetado y compresión

- gzip
- bzip2
- tar

Análisis del sistema

- memtester
- testdisk

Procesado de imágenes

- libpng
- libjpeg

Herramientas de teclado

- kbd (loadkeys, getkeycodes...)

Instalación de nuevas herramientas y aplicaciones

- make
- autoconf
- automake
- libtool
- wget

## 5.3. Pasos a seguir para la instalación del sistema - Versión 1

Para crear un nuevo sistema la opción más sencilla es utilizar una carpeta del sistema, como por ejemplo la carpeta de usuario, y hace que sea la raíz del sistema. En mi caso se llama /home/pfc, así que es en ese directorio donde se instalarán todos los paquetes.

Antes de empezar a distinguir entre versiones conviene aclarar como se hace para tener un compilador que utilice las bibliotecas de C compiladas por nosotros mismos.

Lo primero que hay que hacer es descargarse el compilador gcc y compilarlo en el Ubuntu11.04 que ya teníamos. Una vez hecho esto hay que compilar las bibliotecas de C (glibc) y el paquete de utilidades binutils, utilizando este nuevo compilador que está compilado por nosotros mismos.

Una vez hecho esto, ya tenemos lo suficiente para poder compilar e instalar los paquetes de las diferentes versiones, así que vuelvo a compilar y instalar el compilador gcc

utilizando lo anteriormente mencionado.

El primer paquete necesario que hay que instalar es readline. Este paquete contiene aplicaciones y funciones necesarias para poder editar la línea de comandos en el nuevo sistema.

El siguiente es el paquete bash, que es un *shell*, o intérprete de lenguaje de comandos. Es lo que aparece en los sistemas donde no hay entorno gráfico, o en el caso de que lo haya, lo más parecido es *xterm*. Es lo que se utiliza para hacer lo que se requiera en el sistema sin necesidad del entorno gráfico.

El paquete coreutils contiene utilidades básicas para poder manipular ficheros, texto y el *shell*. Estas son las utilidades básicas que cualquier sistema operativo debe tener.

Una vez está instalado todo lo básico, ya se puede hacer una distinción de la versión 1. Esta versión está destinada principalmente a la reparación y análisis de sistemas operativos locales de otras máquinas.

Procps es un paquete que nos permite ver información sobre los procesos que están corriendo en el sistema.

Findutils contiene utilidades para la búsqueda de archivos y/o directorios.

El paquete util-linux proporciona una serie de utilidades las cuales están orientadas al mantenimiento y la modificación del sistema, como particionar discos duros, crear sistemas de ficheros...

Grep sirve para buscar en el contenido de los archivos el patrón de texto que se le indica.

Less es una utilidad que nos permite, en las ocasiones en que aparece demasiado texto en el *shell* y no nos da tiempo a verlo, poder avanzar o retroceder para poder verlo todo.

El programa vim es un editor de texto simple para línea de comandos. Se pueden ver o editar archivos de texto a través del *shell*.

La utilidad sed sirve para filtrar el texto que se le indica en ficheros y, por ejemplo, reemplazarlo por otro.

Fdisk es un comando que nos permite principalmente ver y modificar las particiones de los discos duros del sistema, y alguna otra cosa más por el estilo.

El paquete tar permite empaquetar varios directorios y/o archivos en uno solo y hacer lo análogo, es decir, desempaquetar.

Gzip es un programa que puede comprimir ficheros para que ocupen menos, ésto hace que no puedan ser usados ni leídos; y descompirmirlos, volviendo a su estado original, para poder volver a utilizarlos.

Bzip es un programa que sirve básicamente para lo mismo que gzip pero utilizando una codificación de compresión diferente.

El paquete grub contiene todo lo necesario para que se lleve a cabo correctamente el arranque del sistema, y también para modificar el arranque.

Net-tools es un paquete que proporciona comandos para el manejo de la red, como por ejemplo levantar una interfaz, poner una dirección IP...

La utilidad ping nos permite enviar paquete a través de la red de los cuales se espera una respuesta y saber así, si una máquina es accesible, o tiene algún tipo de problema de conexión.

El comando tcpdump sirve para monitorizar los paquetes que pasan por el equipo donde se ejecuta, y permite saber cosas como de qué equipo vienen dichos paquetes.

El programa memtester tiene la función, principalmente, de chequear la memoria en busca de fallos.

Man es el comando que permite descodificar las páginas del manual de los paquetes para

poder leerlas.

El paquete testdisk tiene funciones muy útiles como recuperar archivos borrados accidentalmente, recuperar particiones de disco... Básicamente es una herramienta de recuperación.

La distribución de teclado que habrá por defecto en el nuevo sistema será la americana, para poder tener la española instalo el paquete kbd, que contiene las utilidades necesarias para cambiar la distribución de teclado, entre otras.

Con todo esto ya se puede dar por finalizada la versión 1. Para ponerla en marcha hay que copiar todo lo que tenemos en el directorio donde lo hemos instalado, es decir, /home/pfc (excluyendo directorios que ya hubiera como “Descargas” o ,“Escritorio”) en el pen drive. Una vez copiado se instala el grub y se crea el script init que es el que inicia el *shell*.

## 5.4. Pasos a seguir para la instalación del sistema - Versión 2

Procedemos a especificar la versión 2. En esta otra versión, el sistema será mucho más parecido a los sistemas que cualquier persona tiene en su PC. Dispondrá de conexión a internet, *runlevels*, entorno gráfico, posibilidad de instalar más programas... En resumen, se podrá hacer prácticamente lo mismo que en los sistemas corrientes.

El paquete sysVinit servirá para que el sistema disponga de *runlevels* mediante el proceso inicial init. Este proceso controla el arranque y el apagado del sistema.

Lfs-bootscripts contiene lo ficheros de configuración necesarios para el arranque y el apagado de la máquina, que son básicamente scripts distribuidos entre los diferentes *run-levels* que inician y/o apagan determinados servicios.

El paquete sysklogd contiene dos servicios que se encargan de captar los mensajes tales como errores, información útil... Tanto de programas, como del kernel, para que nos sea

más simple encontrar determinados errores.

El paquete udev contiene todo lo necesario para que el sistema pueda gestionar todos los dispositivos. Como por ejemplo recoger la información que den los dispositivos.

Para poder hacer instalaciones des de código fuente (con los típicos pasos, configure+make+make install), a parte de los paquetes básicos que ya están instalados (gcc, glibc...), se requieren 5 paquetes más: make, gawk, autoconf, automake y libtool.

Make es un programa que controla la generación de ejecutables y otros archivos necesarios para una instalación a partir de un fichero “Makefile”, que viene en todos los paquetes en código fuente.

El paquete gawk sirve para extraer información de archivos, o cambiarla, de una manera bastante simple, cosa que es muy útil en las instalaciones ya que se tienen que revisar elementos como los directorios de destino, los archivos a copiar...

Autoconf crea un fichero de configuración para la instalación de paquetes automáticamente sin necesidad de la intervención del usuario.

Automake, como su propio nombre indica, sirve para la generación automática de Makefiles.

Libtool es una biblioteca que facilita la tarea de compartir determinadas bibliotecas, debido a que su función es la de crear bibliotecas portables. Autoconf y automake la utilizan al hacer instalaciones a partir de código fuente.

El paquete module-init-tools contiene programas para manejar módulos del kernel y poder así cargarlos.

Ahora viene la parte más visual, la que sirve para hacer más fácil la compresión de la monitorización de servicios, y también facilitar otras características como puede ser, por ejemplo, la navegación web.

Lo primero a instalar es el servidor X para el entorno de escritorio, XFree86.

El siguiente paso es instalar el gestor de ventanas. Como para lo que se trata en el proyecto no se necesita nada complejo, he optado por FluxBox, es muy liviano y fácil de manejar.

El programa que se va a utilizar para monitorización es el nagios, que usa un servidor web para mostrar información de la monitorización. Por lo tanto, se requiere también un navegador. Como el sistema que tenemos hasta el momento es muy simple, no es recomendable instalar un navegador muy potente como los más actuales, porque tienen muchas dependencias. En lugar de eso, hay que optar por uno más simple. Con éste navegador, cuyo nombre es Dillo, se pueden hacer muchas cosas, pero tenemos la limitación que no tendrá javascript, así que no se puede entrar en páginas como Facebook.

Para poder descargar paquetes en código fuente (o cualquier otro tipo de archivo), se necesita el comando wget.

El paquete apache es el servidor web, que permite que nuestra máquina pueda alojar cosas como archivos, o páginas web, que sean accesibles desde cualquier lugar, o solo desde nuestra máquina.

Por último, el paquete nagios, que es el programa que se encarga de monitorizar los recursos que se le indiquen, como la red, el servidor web... Es un herramienta muy útil que tiene una interfaz muy simple basada en un servidor web, que además permite opciones como avisar por mail cuando algún servicio determinado deja de funcionar, o un equipo deja de ser accesible...

# 6. Diseño

La Ilustración 1 muestra la estructura del sistema de ficheros de la versión 2 (como la versión 1 es una versión reducida de la versión 2, también está incluida) mostrando los directorios más importantes que forman el árbol de directorios, junto con la ubicación general de los paquetes que se van a instalar.

La rama más importante, como se ve a continuación, será /usr/bin, puesto que es ahí donde se van a instalar la mayoría de las aplicaciones.

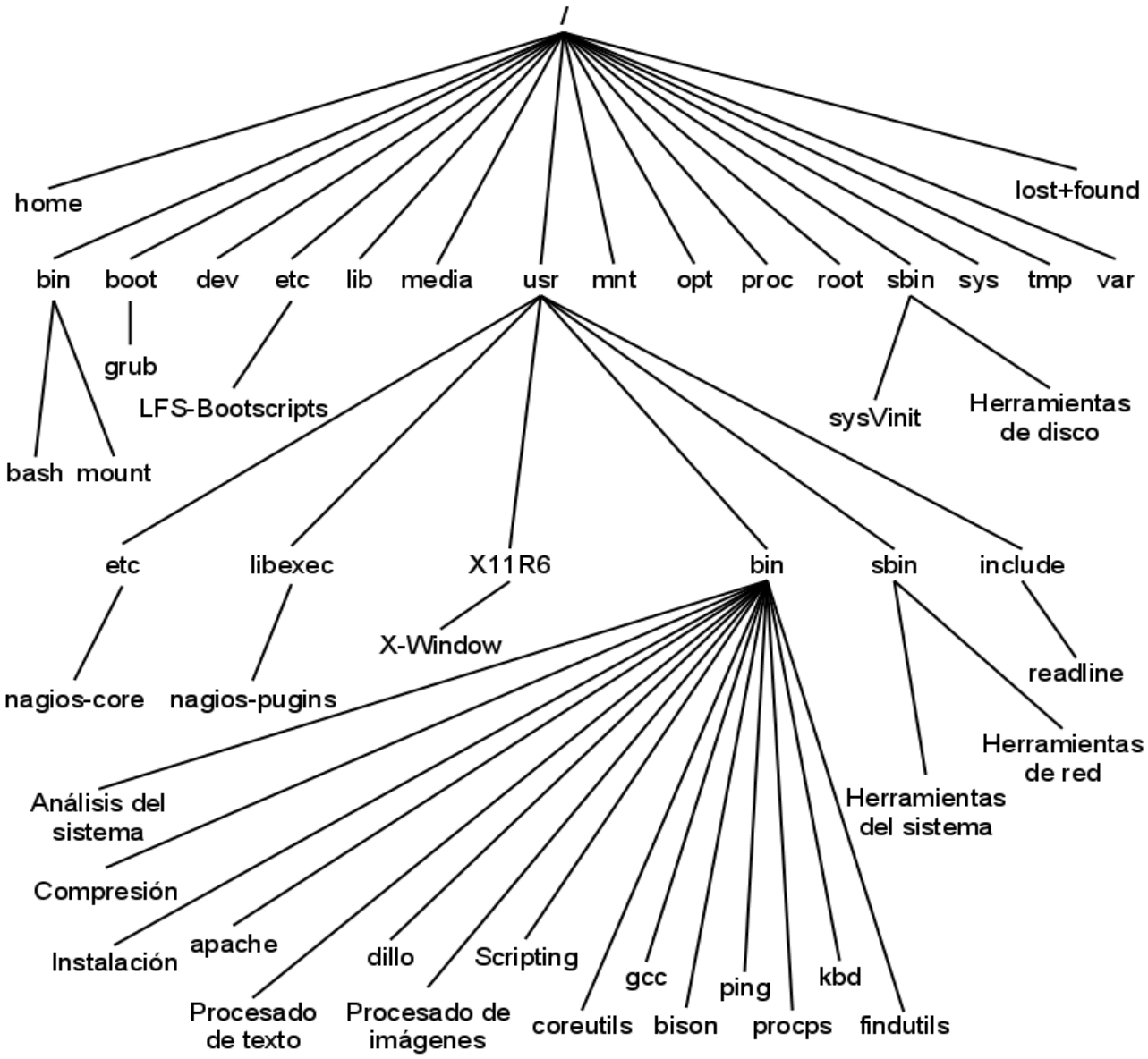



*Ilustración 1: Estructura de directorios y paquetes*

Una vez vista la estructura de directorios y paquetes, voy a explicar lo que deben contener los directorios más relevantes de dicha estructura.

/root: es el directorio de usuario del usuario root o superusuario.
/boot: ficheros que se utilizan para arrancar el sistema (grub, kernel...).
/dev: contiene los dispositivos que hay en el sistema: discos, lectora de CD/DVD, sonido, descriptores de la consola y los terminales, y otros ficheros de dispositivo útiles (/dev/null, /dev/zero, /dev/random...).
/etc: ficheros de configuración de programas que utiliza la máquina.
/bin: contiene los ejecutables que pueden ser usados por la mayoría de usuarios.
/lib: contiene las librerías más básicas utilizadas por todas las herramientas y aplicaciones del sistema (libc, libm...).
/sbin: contiene los ejecutables de administración del sistema, que normalmente sólo pueden ser usados por el superusuario.
/var: ficheros varios, como logs, bases de datos...
/usr: es un directorio que tiene una estructura de subdirectorios parecida a la raíz (/), y contiene utilidades, programas y librerías para los usuarios y el superusuario, que no son necesarios durante el boot del sistema operativo.
/usr/bin: comandos de usuario.
/usr/sbin: comandos para el superusuario.
/usr/lib: librerías adicionales de soporte a las herramientas del sistema y aplicaciones.
/usr/include: contiene las cabeceras, ficheros con las definiciones de las estructuras de datos e interfaz de librerías y sistema operativo, para el compilador.
/usr/X11R6: en éste directorio está todo lo relacionado con el sistema gráfico X-Windows.
/usr/local: directorio para la instalación de herramientas particulares de cada ordenador
/opt: directorio para instalar herramientas y aplicaciones, que en general puede ser compartido entre varios ordenadores.
/home: es donde están los directorios (las cuentas) de todos los usuarios, menos el de root.
/tmp: contiene los ficheros temporales.
/lost+found: contiene archivos que son encontrados cuando se examina el disco en busca de errores.
/media: punto de montaje para dispositivos como pendrives, discos duros, lectores de discos...

/mnt: se utiliza para montar sistemas de ficheros temporales.
/proc: es un sistema de ficheros virtual que tiene información y sucesos del núcleo, de procesos y de cierto hardware.
/sys: tiene información sobre dispositivos proporcionada directamente por el kernel.

## 6.1 Decisiones de diseño

A lo largo del proyecto se han tenido que tomar algunas decisiones de diseño importantes. En este apartado las explicamos. Están relacionadas principalmente con la selección de qué paquetes concretos de software se van a instalar en el sistema, ya que el software disponible es muy grande y habitualmente una misma funcionalidad la pueden suministrar paquetes parecidos. Se trata, pues, de seleccionar los paquetes mas apropiados para nuestro sistema.

### 6.1.1. Herramientas básicas del sistema

Un sistema operativo Linux dispone de numerosos comandos básicos. Para sistemas *embedded*, donde el tamaño de los ficheros puede ser un problema, existe un paquete de software - *busybox* - que ofrece versiones reducidas de la mayoría de comandos. En esta herramienta, sólo se tiene que instalar un comando, *busybox*, que contiene el código de todos los comandos implementados, y crear symbolic links con el nombre de los comandos de Linux (ls, vi, cat...) apuntando al ejecutable de *busybox*. Cuando es invocado, éste comprueba el nombre de fichero que ha utilizado el usuario y, en función del que sea, ejecuta su funcionalidad.

Debido a que en este proyecto no tenemos restricciones de espacio en disco, se ha preferido configurar y compilar los comandos originales, ya sean proporcionados por GNU, o a través de otras fuentes, aprendiendo así cómo se tiene que instalar cada uno de ellos y logrando el máximo parecido a un sistema Linux corriente.

### 6.1.2. Scripts de servicios de arranque y apagado

Para que al arrancar se inicien los servicios necesarios hay que tener algunos ficheros de configuración. Existen dos opciones principales:

1. LFS-Bootscripts se ha obtenido de “linux from scratch”[4]. Es un paquete que contiene todas las utilidades necesarias para iniciar el sistema:

LFS-Bootscripts:
-Network Configuration:
  Script Files:
    rc.d/init.d/network
    sysconfig/network-devices/ifup
    sysconfig/network-devices/ifdown
    sysconfig/network-devices/services/*

  Configuration Files:
    sysconfig/network-devices/ifconfig.*/*

  Additional Configuration:
    sysconfig/network-devices/ifconfig.*

-SetClock configuration:
  Script Files:
    rc.d/init.d/setclock

  Configuration Files:
    sysconfig/clock

-CleanFS configuration:
  Script Files:

rc.d/init.d/cleanfs

Configuration Files:

/etc/sysconfig/createfiles

-Etcétera.

2. La segunda opción[12] consiste en instalar unos paquetes de utilidades de arranque para sysVinit. Estos paquetes son:

- initscripts
- sysv-rc
- sysvinit-utils

Me decanto por la primera opción principalmente porque no existen estos paquetes en código fuente para la arquitectura i386, solo están los binarios, y el objetivo principal del PFC es un sistema basado en código fuente.
Por otro lado, no hay ninguna razón de peso para descartar la primera opción: es software libre y utiliza una estructura muy similar a los paquetes mencionados en la opción 2, igual que otros sistemas de arranque de software libre.

### 6.1.3. Servidor de X-Window

Como en el caso anterior, también hay dos posibles elecciones principales para instalar un servidor X:

1. XFree86[5] es el que se ha utilizado durante muchos años en la mayoría de los sistemas Linux. Hoy en día está descontinuado, aunque como no hace mucho que se dejó de usar, no está excesivamente desfasado.

2. Xorg[6] es el que utilizan actualmente la mayoría de los sistemas Linux y, por lo tanto, está actualizado.

Me decanto por la primera opción, aunque la segunda parece mejor a simple vista. Analizando lo que se necesita para el proyecto, se quiere un sistema para monitorizar otros equipos, por lo que no recibirá actualizaciones de software (o no muchas), tiene una función principal. La primera opción es también mas liviana que la segunda, y que esté descontinuado no es realmente un problema, porque al no estar muy desfasado cumple perfectamente los requisitos necesarios para el sistema.

# 7. Implementación

En esta sección se describe paso a paso la creación del sistema. Se incluye la configuración, compilación e instalación de todas las herramientas, comandos y aplicaciones de usuario. La parte del kernel de Linux se presenta en el Anexo A, dado que se ha completado parcialmente, aproximadamente al 80%. El contenido de esta sección está pensado para poderse reproducir durante una nueva instalación del sistema y como resultado, es una descripción muy técnica. El lector menos interesado en estos detalles de implementación, puede seguir en la siguiente sección (8. Pruebas).

## 7.1. Backups

A lo largo de la descripción se indican los momentos en los que se realizaron los backups. Esto es especialmente importante, ya que el sistema en construcción puede perderse en casos de fallo del ordenador, o simplemente a causa de la instalación incorrecta de alguna herramienta o aplicación. Todos lo backups que se hacen, a menos que se indique lo contrario, es de la máquina virtual, no solo de los archivos del nuevo sistema. Esto hace que sea más lento hacer los backups, pero en caso de pérdida de los datos, es mucho más fácil y rápido restaurar la máquina virtual que tener que crearla de nuevo y copiar los ficheros del nuevo sistema.

## 7.2. Proceso de instalación

Se empieza, usando virtualbox, con la creación de una máquina virtual, para instalar Ubuntu11.04 x86, con un espacio de almacenamiento de 14 GB y 1,5 GB de RAM. La máquina virtual dispondrá de las siguientes particiones:

- / 13 GB Primaria
- swap 1 GB Primaria

Una vez instalado Ubuntu11.04, se crea un único usuario del sistema: “pfc”. Este usuario será el que dará soporte a la instalación de todos los paquetes de nuestras versiones 1 y 2.

A partir del directorio de entrada del usuario “pfc” (/home/pfc), se crea la estructura de directorios del nuevo sistema. El objetivo es que /home/pfc/ sea a partir de ahora una imagen del directorio raíz (con todos los subdirectorios necesarios) del nuevo sistema. Para crear cada directorio se utiliza el comando mkdir:

```
mkdir /home/pfc/usr;
mkdir /home/pfc/boot;
mkdir /home/pfc/dev;
mkdir /home/pfc/etc;
mkdir /home/pfc/home;
mkdir /home/pfc/bin;
mkdir /home/pfc/lib;
mkdir /home/pfc/media;
mkdir /home/pfc/mnt;
mkdir /home/pfc/opt;
mkdir /home/pfc/proc;
mkdir /home/pfc/root;
mkdir /home/pfc/sbin;
mkdir /home/pfc/sys;
mkdir /home/pfc/tmp;
mkdir /home/pfc/var;
```

Una vez se vayan instalando comandos en esta imagen, queremos que nuestro sistema los utilice por defecto. De esta forma, si se instala una versión de un comando más nueva que los comandos que tiene el Ubuntu11.04 de base, no habrá ningún problema de incompatibilidad. También así, nos aseguramos que los nuevos comandos instalados funcionan correctamente. Para hacerlo, debemos comprobar que la variable PATH los contiene en primera posición. Podemos comprobarlo con el comando:

```
echo $PATH
```

Si falta alguno de los directorios de binarios, hay que añadirlos. Por ejemplo, el siguiente ejemplo muestra como añadirlos todos:

```
export PATH=/home/pfc/bin:\
            /home/pfc/sbin:\
            /home/pfc/usr/bin:\
            /home/pfc/usr/sbin:\
            $PATH
```

Los nuevos componentes para la variable PATH deben colocarse al principio porque al buscar los programas se hará en el orden de los directorios indicados. Al reiniciar, estos cambios se perderán, así que habrá que ejecutar el comando anterior cada vez que se reinicie la máquina o se inicie una nueva sesión. En esta versión del sistema, al arrancar el intérprete de comandos bash, como si fuera el proceso init, no lee los ficheros de entrada .profile o .basrc, con lo cual no se puede automatizar el cambio de la variable PATH.

### 7.2.1. Programas básicos necesarios

Los programas básicos necesarios para el uso de un sistema cualquiera son:

- GNU:
  - Gcc
  - GMP
  - MPC
  - MPFR
  - Glibc
  - Binutils
  - Readline
  - Bash
  - Coreutils

Por lo tanto éstos son los que se deben instalar antes de nada. A continuación se explica como hacerlo de manera correcta, ya que si antes de tener el compilador, se instala otra cosa, ya no estará compilado por nuestro propio compilador.

Para empezar, se instala por primera vez el compilador gcc:

Gcc

Descarga: http://gcc.gnu.org/releases.html

Descomprimir: ~/Descargas$ tar jxvf gcc-4.6.1.tar.bz2;

Dentro de la carpeta INSTALL hay información sobre la instalación[1]:

**Prerequisitos**

Se necesita gmp, mpfr, mpc: los instalaremos a la vez que el compilador, lo explicaré más adelante.

Descarga:
gmp-5.0.2: http://gmplib.org/
mpfr-3.0.1: http://www.mpfr.org/mpfr-current/
mpc-0.9: http://www.multiprecision.org/index.php?prog=mpc&page=download

Descomprimir:
~/Descargas$ tar jxvf gmp-5.0.2.tar.bz2;
~/Descargas$ tar jxvf mpfr-3.0.1.tar.bz2;
~/Descargas$ tar zxvf mpc-0.9.tar.gz;

Para instalarlos al mismo tiempo que el compilador, hacemos symbolic links dentro del directorio gcc-4.6.1 de:

- gmp: ~/Descargas/gcc-4.6.1$ ln -s ../gmp-5.0.2 gmp
- mpfr: ~/Descargas/gcc-4.6.1$ ln -s ../mpfr-3.0.1 mpfr

- mpc: ~/Descargas/gcc-4.6.1$ ln -s ../mpc-0.9 mpc

Configuración:
~/Descargas/gcc-4.6.1$ ./configure

Antes de compilar hay que instalar gcc-multilib, porque hay un bug en Ubuntu11.04, y con esta instalación se crea un symbolic link que restaura /usr/include/linux/ al mismo estado que en Ubuntu 10.10:
sudo apt-get install gcc-multilib;

No se sabe porque pasa ésto exactamente, pero se cree que es por algo relacionado con los compiladores intel y/o los procesadores de 64 bits (que es el que tengo yo, aunque el sistema operativo que use sea de 32 bits).

Compilación:
~/Descargas/gcc-4.6.1$ make;

Tarda aproximadamente 1h.

Instalación:
~/Descargas/gcc-4.6.1$ sudo make install;

Con este compilador provisional compilado por nosotros, vamos compilar el paquete glibc-2.13 ya en el nuevo sistema (/home/pfc)

Glibc[2]

Descarga:
glibc-2.13: http://ftp.gnu.org/gnu/glibc/
Y también el parche para eliminar el error que trae:
glibc-2.13-gcc_fix-1.patch: http://lfs-matrix.net/patches/downloads/glibc/

Descomprimir:

~/Descargas$ tar jxvf glibc-2.13.tar.bz2;

Aplico el parche:
~/Descargas/glibc-2.13$ patch -Np1 -i ../glibc-2.13-gcc_fix-1.patch

Para compilarlo hace falta versiones recientes de los paquetes gcc y make.
Para configurarlo hay que hacerlo en un directorio separado al que hemos desempaquetado, así que creo el directorio ~/Descargas/glibc_build. En el fichero “INSTALL“ están las opciones necesarias ara la configuración.

Antes de configurarlo instalamos los paquetes necesarios:
sudo apt-get install texinfo;
sudo apt-get install autoconf;
sudo apt-get install gawk;

Hay un problema de nombres en las cabeceras del sistema debido al cambio de versiones, hay que hacer un symbolic link a la carpeta correcta:
sudo ln -s /usr/src/linux-headers-2.6.38-8/arch/x86/include/asm/ /usr/src/linux-headers-2.6.38-8/include/asm;

Configuración:
~/Descargas/glibc_build$ ../glibc-2.13/configure --prefix=/usr --with-headers=/usr/src/linux-headers-2.6.38-8/include/ --enable-add-ons >glibc_configure.txt

Compilación:
~/Descargas/glibc_build$ make > glibc_make.txt

Instalación:
~/Descargas/glibc_build$ make install_root=/home/pfc/ install;

Ya tenemos instalado glibc-2.13 en el nuevo sistema. Ahora hay que copiar las bibliotecas necesarias para que todo funcione correctamente:
cp -r /usr/src/linux-headers-2.6.38-8/include/asm-generic /home/pfc/usr/include/;
cp -r /usr/src/linux-headers-2.6.38-8/include/linux /home/pfc/usr/include/;

cp -r /usr/src/linux-headers-2.6.38-8/arch/x86/include/asm /home/pfc/usr/include/;

Ya tenemos el nuevo sistema con las bibliotecas así que ya se puede instalar el compilador gcc-4.6.1:

Como la primera vez, instalamos gmp, mpfr, mpc a la vez que el compilador. Esta vez, que será el compilador que tendremos definitivamente en el nuevo sistema, instalamos también binutils junto el compilador.

Descargar:
binutils-2.21: http://ftp.gnu.org/gnu/binutils/

Descomprimir:
~/Descargas$ tar jxvf binutils-2.21.tar.bz2;

Para evitar posibles problemas, hay que borrar el directorio donde está gcc preparado para instalar y volverlo a desempaquetar.

Para instalar binutils a la vez que el compilador hay que hacer symbolic links dentro del directorio gcc-4.6.1 de todo lo que hay dentro del directorio binutils:
~/Descargas/gcc-4.6.1$ for file in ../binutils-2.21/* ; do ln -s "${file}" ; done

Configuración:
~/Descargas/gcc-4.6.1$ ./configure --prefix=/usr --with-sysroot=/home/pfc --with-build-sysroot=/home/pfc --enable-languages=c,c++,fortran > gcc_configure.txt;

Antes de compilar hay que ir al fichero /home/pfc/usr/include/linux/types.h, y comentar la línea que dice “warning "Attempt to use kernel headers from user space, see http://kernelnewbies.org/KernelHeaders"”. Esta línea la trata como un *warning* y al tener el flag -Werror activado, al haber muchos *warnings* da error.

Compilación:
~/Descargas/gcc-4.6.1$ make > gcc_make.txt;

Instalación:
~/Descargas/gcc-4.6.1$ sudo make DESTDIR=/home/pfc install;

Readline

Descarga:
readline-6.2: http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html

Descomprimir:
~/Descargas$ tar xvfz readline-6.2.tar.gz

Dentro de la carpeta doc está información sobre cómo utilizar el programa o cómo programarlo.
En el fichero INSTALL está la información necesaria para hacer el configure y el make.

Configuración:
~/Descargas/readline-6.2$ ./configure --prefix=/usr CC=/home/pfc/usr/bin/gcc RANLIB=/home/pfc/usr/bin/ranlib AR=/home/pfc/usr/bin/ar > configure_readline.txt

Compilación:
~/Descargas/readline-6.2$ make > make_readline.txt

Instalación:
~/Descargas/readline-6.2$ make DESTDIR=/home/pfc install

Bash

Descarga:
bash-4.2: http://ftp.gnu.org/gnu/bash/

Descomprimir:
~/Descargas$ tar xvfz bash-4.2.tar.gz

Configuración:
~/Descargas/bash-4.2$ ./configure --prefix=/usr --enable-readline --with-installed-readline=/home/pfc/usr/lib CC=/home/pfc/usr/bin/gcc RANLIB=/home/pfc/usr/bin/ranlib AR=/home/pfc/usr/bin/ar > configure_bash.txt

Compilación:
~/Descargas/bash-4.2$ make > make_bash.txt

Instalación:
~/Descargas/bash-4.2$ make DESTDIR=/home/pfc install

Coreutils

Descarga:
coreutils-8.12: http://ftp.gnu.org/gnu/coreutils/

Descomprimir:
~/Descargas$ tar xvfz coreutils-8.12.tar.gz

Configuración:
~/Descargas/coreutils-8.12$ ./configure --prefix=/usr --with-gnu-ld CC="/home/pfc/usr/bin/gcc -std=gnu99" CPP="/home/pfc/usr/bin/gcc -std=gnu99 -E" RANLIB=/home/pfc/usr/bin/ranlib > configure_coreutils.txt

Compilación:
~/Descargas/coreutils-8.12$ make > make_coreutils.txt

Instalación:
~/Descargas/coreutils-8.12$ sudo make DESTDIR=/home/pfc install

Cambio la raíz para comprobar que funciona el nuevo sistema:
sudo chroot /home/pfc;

Nos indica que no existe el comando /bin/bash, necesario para lanzar la consola de comandos del nuevo sistema. Esto sucede porque se ha instalado bash en /usr, por tanto el ejecutable está en /usr/bin. Para solucionarlo únicamente hay que hacer un symbolic link del ejecutable al lugar donde lo busca:
~/usr/bin/$ ln -s ./bash ../../bin/

Ahora se puede volver a ejecutar:
sudo chroot /home/pfc;

Y ya estamos dentro del nuevo sistema. Se puede comprobar que los comandos básicos instalados por coreutils funcionan perfectamente.

//////////////////////////////////////////////////backup 25-09-2011//////////////////////////////////////////

Ahora mismo tenemos un sistema mínimo que funciona, pero que en la práctica sirve de muy poco, así que hay que preparar el sistema para alguna cosa en especial.

Se proponen dos versiones que servirán para diferentes cosas:

- Versión 1:

En esta versión se preparará el sistema para que sirva en caso de que el SO local de una máquina nuestra se estropee, nos ayude a arreglarlo (como por ejemplo eliminación accidental del grub, sectores del disco defectuosos...), y lo pondremos como una imagen en un pendrive para iniciarlo directamente al encender la máquina.

- Versión 2:

En esta versión el nuevo sistema será similar a un SO de los que tiene cualquier persona instalado en su PC. Tendrá runlevels y conexión a internet.

Procedemos a hacer las distintas versiones:

### 7.2.2. Versión 1

Programas necesarios

Hay que instalar programas que serán necesarios para poder arreglar un sistema:

- GNU:
    - Procps
    - Ncurses
    - Findutils
    - Grep
    - Less
    - Vim
    - Sed
    - Fdisk
    - Tar
    - Gzip
    - Grub
    - Bison
    - Flex
    - Tcpdump
    - Libcap
    - Memtester
    - Man
    - Groff
    - Testdisk

- No-GNU
    - Util-linux
    - Bzip2
    - Net-tools
    - Ping

- Kbd

Procps

Dependencias: Ncurses

Descarga:
procps-3.2.8: http://procps.sourceforge.net/index.html

Descomprimir:
~/Descargas$ tar xvfz procps-3.2.8.tar.gz

Ya está el make hecho así que no necesitamos hacer el configure, le doy la configuración directamente en el make install. Pero antes hay que cambiar el directorio de destino de las bibliotecas, porque el nuevo sistema es de 32 bits. En el Makefile hay que modificar:
lib64 := lib$(shell [ -d /lib64 ] && echo 64) ----------> lib64 := lib

Compilación:
~/Descargas/procps-3.2.8$ make CC=/home/pfc/usr/bin/gcc PREFIX=/usr DESTDIR=/home/pfc install

Instalación:
~/Descargas/procps-3.2.8$ sudo make CC=/home/pfc/usr/bin/gcc PREFIX=/usr DESTDIR=/home/pfc install

Ncurses

Descarga:
ncurses-5.7: http://invisible-island.net/ncurses/ncurses.html#download_ncurses

Descomprimir:
~/Descargas$ tar xvfz ncurses.tar.gz

Configuración

~/Descargas/ncurses-5.7-20090718$ ./configure --prefix=/usr CC=/home/pfc/usr/bin/gcc RANLIB=/home/pfc/usr/bin/ranlib AR=/home/pfc/usr/bin/ar

Compilación:
~/Descargas/ncurses-5.7-20090718$ make

Instalación:
~/Descargas/ncurses-5.7-20090718$ sudo make DESTDIR=/home/pfc install

Findutils

Descarga:
findutils-4.4.2: http://ftp.gnu.org/pub/gnu/findutils/

Descomprimir:
~/Descargas$ tar xvfz findutils-4.4.2.tar.gz

Configuración:
~/Descargas/findutils-4.4.2$ ./configure --prefix=/usr CC="/home/pfc/usr/bin/gcc -std=gnu99" CPP="/home/pfc/usr/bin/gcc -E" RANLIB=/home/pfc/usr/bin/ranlib

Compilación:
~/Descargas/findutils-4.4.2$ make

Instalación:
~/Descargas/findutils-4.4.2$ sudo make DESTDIR=/home/pfc install

Util-linux

Descarga:
util-linux_2.19.1-2: http://bouyguestelecom.ubuntu.lafibre.info/ubuntu/pool/main/u/util-linux/

Descomprimir:
~/Descargas$ xvfz util-linux_2.19.1-2ubuntu3.tar.gz

Configuración:
~/Descargas/util-linux.ubuntu$ ./configure --prefix=/usr --enable-login-utils --without-ncurses CC="/home/pfc/usr/bin/gcc -std=gnu99" CPP="/home/pfc/usr/bin/gcc -std=gnu99 -E" RANLIB=/home/pfc/usr/bin/ranlib AR=/home/pfc/usr/bin/ar

Antes de compilar hay que copiar unas bibliotecas del código fuente descargado al nuevo sistema, que son necesarias para algunos de los comandos que se instalan:
~/Descargas/util-linux.ubuntu$ cp shlibs/uuid/src/.libs/libuuid.so.1 /home/pfc/usr/lib/
~/Descargas/util-linux.ubuntu$ cp shlibs/blkid/src/.libs/libblkid.so.1 /home/pfc/usr/lib/

También hay que abrir el fichero ~/Descargas/util-linux.ubuntu/mount/loop.h y sustituir la línea “#define my_dev_t __kernel_dev_t” por esta otra: “#define my_dev_t dev_t”. Porque en el fichero donde está declarada (/usr/src/linux-headers-2.6.38-8-generic/include/linux/types.h) le pone otro nombre a la variable: “typedef __kernel_dev_t **dev_t**;”

Compilación:
~/Descargas/util-linux.ubuntu$ make

Instalación:
~/Descargas/util-linux.ubuntu$ sudo make DESTDIR=/home/pfc install

Grep

Descarga:
grep-2.9: ftp://mirror.cict.fr/gnu/grep/

Descomprimir:
~/Descargas$ tar xvfz grep-2.9.tar.gz

Configuración:
~/Descargas/grep-2.9$ ./configure --prefix=/usr CC="/home/pfc/usr/bin/gcc -std=gnu99" CPP="/home/pfc/usr/bin/gcc -E" RANLIB=/home/pfc/usr/bin/ranlib

Compilación:
~/Descargas/grep-2.9$ make

Instalación:
~/Descargas/grep-2.9$ sudo make DESTDIR=/home/pfc install

Less

Descarga:
less-444: http://www.greenwoodsoftware.com/less/index.html

Descomprimir:
~/Descargas$ tar xvfz less-444.tar.gz

Configuración:
~/Descargas/less-444$ ./configure --prefix=/usr CC=/home/pfc/usr/bin/gcc

Compilación:
~/Descargas/less-444$ make

Instalación:
~/Descargas/less-444$ sudo make DESTDIR=/home/pfc install

Vim

Descarga:
vim-7.3: ftp://ftp.vim.org/pub/vim/unix/

Descomprimir:
~/Descargas$ tar xvfj vim-7.3.tar.bz2

Configuración:
~/Descargas/vim73$ ./configure --prefix=/usr CC=/home/pfc/usr/bin/gcc

Compilación:
~/Descargas/vim73$ make

Instalación:
~/Descargas/vim73$ sudo make DESTDIR=/home/pfc install

Sed

Descarga:
sed-4.2.1: http://ftp.gnu.org/gnu/sed/

Descomprimir:
~/Descargas$ tar xvfj sed-4.2.1.tar.bz2

Configuración:
~/Descargas/sed-4.2.1$ ./configure --prefix=/usr CC="/home/pfc/usr/bin/gcc" CPP="/home/pfc/usr/bin/gcc -E" RANLIB=/home/pfc/usr/bin/ranlib

Compilación:
~/Descargas/sed-4.2.1$ make

Instalación:
~/Descargas/sed-4.2.1$ sudo make DESTDIR=/home/pfc install

Fdisk

Descarga:
fdisk-1.3.0a: ftp://ftp.gnu.org/gnu/fdisk/

Descomprimir:
~/Descargas$ tar xvfj fdisk-1.3.0a.tar.bz2

Configuración:

~/Descargas/fdisk-1.3.0a$ ./configure --prefix=/usr --with-gnu-ld CC="/home/pfc/usr/bin/gcc -std=gnu99" CPP="/home/pfc/usr/bin/gcc -std=gnu99 -E" RANLIB=/home/pfc/usr/bin/ranlib > configure_coreutils.txt

Compilación:
~/Descargas/fdisk-1.3.0a$ make > make_coreutils.txt

Instalación:
~/Descargas/fdisk-1.3.0a$ sudo make DESTDIR=/home/pfc install

Tar

Descarga:
tar-1.26: ftp://ftp.gnu.org/gnu/tar/

Descomprimir:
~/Descargas$ tar xvfj tar-1.26.tar.bz2

Configuración:
~/Descargas/tar-1.26$ ./configure --prefix=/usr CC="/home/pfc/usr/bin/gcc -std=gnu99" CPP="/home/pfc/usr/bin/gcc -E" RANLIB=/home/pfc/usr/bin/ranlib

Compilación:
~/Descargas/tar-1.26$ make

Instalación:
~/Descargas/tar-1.26$ sudo make DESTDIR=/home/pfc install

Gzip

Descarga:
gzip-1.4: http://ftp.gnu.org/gnu/gzip/

Descomprimir:

~/Descargas$ tar xvfz gzip-1.4.tar.gz

Configuración:
~/Descargas/gzip-1.4$ ./configure --prefix=/usr CC="/home/pfc/usr/bin/gcc -std=gnu99" CPP="/home/pfc/usr/bin/gcc -std=gnu99 -E" RANLIB=/home/pfc/usr/bin/ranlib

Compilación:
~/Descargas/gzip-1.4$ make

Instalación:
~/Descargas/gzip-1.4$ sudo make DESTDIR=/home/pfc install

Bzip2

Descarga:
bzip2-1.0.6: http://www.bzip.org/downloads.html

Descomprimir:
~/Descargas$ tar xvfz bzip2-1.0.6.tar.gz

Ya tiene el Makefile hecho, así que hay que hacer el make install configurándolo anteriormente a mano. Hay que abrir el fichero Makefile y cambiar las variables de configuración:

- CC=/home/pfc/usr/bin/gcc
- AR=/home/pfc/usr/bin/ar
- RANLIB=/home/pfc/usr/bin/ranlib
- PREFIX=/home/pfc/usr

Instalación:
~/Descargas/bzip2-1.0.6$ sudo make install;

Grub

Dependencias: Bison, Flex

Descarga:
grub-1.99: ftp://ftp.gnu.org/gnu/grub/

Descomprimir:
~/Descargas$ tar xvfz grub-1.99.tar.gz

Hay que añadir /home/pfc/usr/bin a la variable PATH para que encuentre el programa bison:
export PATH=/home/pfc/usr/bin:$PATH;

Configuración:
~/Descargas/grub-1.99$ ./configure --prefix=/usr CC=/home/pfc/usr/bin/gcc CPP="/home/pfc/usr/bin/gcc -E" RANLIB=/home/pfc/usr/bin/ranlib

Antes de hacer el make copiamos unos archivos relacionados con bison para solucionar un problema de manera rápida, y es que busca en /usr/share en vez de en /home/pfc/usr/share:
sudo mkdir /usr/share/bison;
sudo mkdir /usr/share/bison/m4sugar;
sudo cp -r /home/pfc/usr/share/bison/m4sugar/m4sugar.m4 \
/usr/share/bison/m4sugar/m4sugar.m4;
sudo cp -r /home/pfc/usr/share/bison/m4sugar/foreach.m4 /usr/share/bison/m4sugar/;
sudo cp -r /home/pfc/usr/share/bison/bison.m4 /usr/share/bison/;
sudo cp -r /home/pfc/usr/share/bison/c-skel.m4 /usr/share/bison/;
sudo cp -r /home/pfc/usr/share/bison/yacc.c /usr/share/bison/;
sudo cp -r /home/pfc/usr/share/bison/c.m4 /usr/share/bison/;

Compilación:
~/Descargas/grub-1.99$ make

Instalación:
~/Descargas/grub-1.99$ sudo make DESTDIR=/home/pfc install

Bison

Descarga:
bison-2.5: http://ftp.gnu.org/gnu/bison/

Descomprimir:
~/Descargas$ tar xvfj bison-2.5.tar.bz2

Configuración:
~/Descargas/bison-2.5$ ./configure --prefix=/usr CC="/home/pfc/usr/bin/gcc -std=gnu99" CPP="/home/pfc/usr/bin/gcc -std=gnu99 -E" CXX=/home/pfc/usr/bin/g++ RANLIB=/home/pfc/usr/bin/ranlib

Compilación:
~/Descargas/bison-2.5$ make

Instalación:
~/Descargas/bison-2.5$ sudo make DESTDIR=/home/pfc install

Flex

Descarga:
flex-2.5.35: http://flex.sourceforge.net/#downloads

Descomprimir:
~/Descargas$ tar xvfj flex-2.5.35.tar.bz2

Configuración:
~/Descargas/flex-2.5.35$ ./configure --prefix=/usr AR=/home/pfc/usr/bin/ar CC=/home/pfc/usr/bin/gcc CPP="/home/pfc/usr/bin/gcc -E" CXX=/home/pfc/usr/bin/g++ RANLIB=/home/pfc/usr/bin/ranlib

Compilación:
~/Descargas/flex-2.5.35$ make

Instalación:
~/Descargas/flex-2.5.35$ sudo make DESTDIR=/home/pfc install

//////////////////////////////////////////////////backup 30-09-2011//////////////////////////////////////////

Net-tools

Descarga:
net-tools-1.60: http://packages.ubuntu.com/source/natty/net-tools

Descomprimir:
~/Descargas$ tar xvfz net-tools_1.60.orig.tar.gz

La configuración ya está hecha, así que hay que abrir el fichero Makefile y cambiar lo necesario:
CC = /home/pfc/usr/bin/gcc

Hay que corregir unos errores que hay en la línea 107 del fichero ~/Descargas/net-tools-1.60/lib/inet_sr.c, y en las líneas 100, 119 y 176 del fichero ~/Descargas/net-tools-1.60/hostname.c, ya que hay un default de un switch que no hace nada. Comentándolos ya se soluciona el problema.

También hay un error en el paso de parámetros de una función del fichero ~/Descargas/net-tools-1.60/lib/x25_sr.c en la línea 80, donde pasa un struct sin indicar que lo es. Hay que cambiar “sizeof(x25_address)”, por “sizeof(struct x25_address)”.

Instalación:
~/Descargas/net-tools-1.60$ sudo make BASEDIR=/home/pfc install;
Cuando pida la configuración hay que dejarlo todo por defecto (dándole al “enter” en cada opción).

Ping

Descargaremos un paquete que, además del comando ping, también trae otras utilidades, pero solo nos interesa el comando ping.

Descarga:
iputils: http://packages.ubuntu.com/source/oneiric/iputils

Descomprimir:
~/Descargas$ tar xvfz iputils_20101006.orig.tar.gz

Ya viene el Makefile hecho, así que compilamos:
~/Descargas/iputils$ make ping

Instalación:
~/Descargas/iputils$ cp ./ping /home/pfc/usr/bin

El comando ping tiene unos permisos especiales:
chmod go-rw /home/pfc/usr/bin/ping;
chmod u+s /home/pfc/usr/bin/ping;

Para que funcione ping necesitamos el protocolo ICMP, que no tenemos instalado. De momento creamos el archivo /home/pfc/etc/protocols con este contenido:

```
#
# protocols     This file describes the various protocols that are
#               available from the TCP/IP subsystem.  It should be
#               consulted instead of using the numbers in the ARPA
#               include files, or, worse, just guessing them.
#

ip      0       IP      # internet protocol, pseudo protocol number
icmp    1       ICMP    # internet control message protocol
igmp    2       IGMP    # internet group multicast protocol
ggp     3       GGP     # gateway-gateway protocol
tcp     6       TCP     # transmission control protocol
```

```
pup     12      PUP     # PARC universal packet protocol
udp     17      UDP     # user datagram protocol
raw     255     RAW     # RAW IP interface

# End.
```

Tcpdump

Dependencias: Libcap

Descarga:
tcpdump-4.1.1: http://www.tcpdump.org/

Descomprimir:
~/Descargas$ tar xfvz tcpdump-4.1.1.tar.gz

Configuración:
~/Descargas/tcpdump-4.1.1$ ./configure --prefix=/usr CC=/home/pfc/usr/bin/gcc RANLIB=/home/pfc/usr/bin/ranlib

Compilación:
~/Descargas/tcpdump-4.1.1$ make

Instalación:
~/Descargas/tcpdump-4.1.1$ sudo make DESTDIR=/home/pfc install

Libcap

Descarga:
libcap-1.1.1: http://www.tcpdump.org/

Descomprimir:
~/Descargas$ tar xvfz libpcap-1.1.1.tar.gz

Configuración:

~/Descargas/libpcap-1.1.1$ ./configure --prefix=/usr CC=/home/pfc/usr/bin/gcc RANLIB=/home/pfc/usr/bin/ranlib LD=/home/pfc/usr/bin/ld

Compilación:

~/Descargas/libpcap-1.1.1$ make

Instalación:

~/Descargas/libpcap-1.1.1$ sudo make DESTDIR=/home/pfc install

Memtester

Descarga:

memtester-4.2.2: http://pyropus.ca/software/memtester/

Descomprimir:

~/Descargas$ tar xvfz memtester-4.2.2.tar.gz

Ya tenemos el configure hecho así que hay que compilar directamente:

~/Descargas/gzip-1.4$ make

Ahora hay que poner el nuevo sistema como lugar donde instalar, abrimos el Makefile y cambiamos:

INSTALLPATH = /home/pfc/usr

Hay que crear el directorio para el manual:

~$ sudo mkdir usr/man/man8

Instalación:

~/Descargas/gzip-1.4$ sudo make install

Man

Dependencias: cat, less, gtbl, nroff

Descarga:
man-1.6g: http://primates.ximian.com/~flucifredi/man/

Descomprimir:
~/Descargas$ tar xvfz man-1.6g.tar.gz

Configuración:
~/Descargas/man-1.6g$ ./configure

Compilación:
~/Descargas/man-1.6g$ make

Instalación:
~/Descargas/man-1.6g$ sudo make DESTDIR=/home/pfc install

//////////////////////////////////////////////////backup 25-10-2011//////////////////////////////////////////

Ya están instalados tanto cat como less, pero tienen que estar en /bin, por lo tanto hay que hacer symbolic links:
~/bin$ ln -s ../usr/bin/cat cat
~/bin$ ln -s ../usr/bin/less less

Groff (contiene nroff y gtbl)

Descarga:
groff-1.21: ftp://ftp.gnu.org/gnu/groff/

Descomprimir:
~/Descargas$ tar xvfz groff-1.21.tar.gz

Configuración:
~/Descargas/groff-1.21$ ./configure --prefix=/usr CC=/home/pfc/usr/bin/gcc CCC=/home/pfc/usr/bin/g++ RANLIB=/home/pfc/usr/bin/ranlib

Compilación:
~/Descargas/groff-1.21$ make

Instalación:
~/Descargas/groff-1.21$ sudo make DESTDIR=/home/pfc install

Ahora se ve que en /home/pfc/usr/bin no está el programa gtbl, esto pasa porque gtbl es simplemente un puntero a tbl, así que hacemos un symbolic link:
~/usr/bin$ ln -s tbl ./gtbl

TestDisk

Descarga:
testdisk-6.12: http://www.cgsecurity.org/wiki/TestDisk_Download

Descomprimir:
~/Descargas$ tar xvfj testdisk-6.12.tar.bz2

Configuración:
~/Descargas/testdisk-6.12$ ./configure CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/libexec --prefix=/usr CC=/home/pfc/usr/bin/gcc

Compilación:
~/Descargas/testdisk-6.12$ make

Instalación:
~/Descargas/testdisk-6.12$ sudo make DESTDIR=/home/pfc install

Por defecto la distribución del teclado es la americana, así que hay que poner la distribución del teclado español. Se necesita el programa loadkeys junto con alguno más que vienen incluidos en el paquete kbd.

Kbd

Descarga:
kbd-1.15.3: http://fossies.org/linux/misc/kbd-1.15.3.tar.gz/

Descomprimir:
~/Descargas$ tar xvfz kbd-1.15.3.tar.gz

Configuración:
~/Descargas/kbd-1.15.3$ ./configure --prefix=/usr CC=/home/pfc/usr/bin/gcc CPP="/home/pfc/usr/bin/gcc -E"

Antes de compilar comentamos unas líneas del fichero ~/Descargas/kbd-1.15.3/es.po que utilizan un comando que no está en el sistema para hacer una traducción, ya que no es algo importante:

- #msgid "Saved %d-char %dx%d font file on %s\n" [línea 1365]
- #msgstr "Se ha guardado el fichero de tipos %2$dx%3$d de %1$d caracteres en %s\n" [línea 1366]

Compilación:
~/Descargas/kbd-1.15.3$ make

Instalación:
~/Descargas/kbd-1.15.3$ sudo make DESTDIR=/home/pfc install

La distribución del teclado en español está en:
./usr/share/keymaps/i386/qwerty/es.map.gz

Para cargarla una vez se haya arrancado el nuevo sistema hay que ejecutar:
loadkeys /usr/share/keymaps/i386/qwerty/es.map.gz
o bien:
loadkeys es

//////////////////////////////////////////////////backup 5-11-2011//////////////////////////////////////////

Con todo lo que tenemos ya se puede montar el sistema en un pendrive con, como mínimo 600MiB. Para poder montarlo, lo primero de todo es formatearlo y darle formato de ficheros ext4. De nombre le ponemos PFC. Al formatearlo crea el directorio lost+found.

Ahora hay que copiar todo el nuevo sistema que está en /home/pfc excluyendo los directorios que no pertenecen al sistema (Descargas, Escritorio...) en el pendrive (/media/PFC).

Para preparar el grub hay que ver como están organizados los discos:
~$ sudo fdisk -l;
Disco /dev/sda: 15.0 GB, 15032385536 bytes
255 cabezas, 63 sectores/pista, 1827 cilindros
Unidades = cilindros de 16065 * 512 = 8225280 bytes
Tamaño de sector (lógico / físico): 512 bytes / 512 bytes
Tamaño E/S (mínimo/óptimo): 512 bytes / 512 bytes
Identificador de disco: 0x0005bd0e

| Dispositivo | Inicio | Comienzo | Fin | Bloques | Id | Sistema |
|---|---|---|---|---|---|---|
| /dev/sda1 | * | 1 | 1704 | 13678592 | 83 | Linux |
| /dev/sda2 | | 1704 | 1828 | 999424 | 82 | Linux swap / Solaris |

Disco /dev/sdb: 3951 MB, 3951034368 bytes
122 cabezas, 57 sectores/pista, 1109 cilindros
Unidades = cilindros de 6954 * 512 = 3560448 bytes
Tamaño de sector (lógico / físico): 512 bytes / 512 bytes
Tamaño E/S (mínimo/óptimo): 512 bytes / 512 bytes
Identificador de disco: 0x000c5b2f

| Dispositivo | Inicio | Comienzo | Fin | Bloques | Id | Sistema |
|---|---|---|---|---|---|---|
| /dev/sdb1 | | 2 | 1110 | 3854336 | b | W95 FAT32 |

Se ve que el pendrive(4GB) es /dev/sdb, por lo tanto es ahí donde hay que instalar el grub.

Primero montamos los directorios temporalmente en /mnt:
sudo mount /dev/sdb1 /mnt
sudo mount --bind /proc /mnt/proc
sudo mount --bind /dev /mnt/dev
sudo mount --bind /sys /mnt/sys

Y ahora accedemos al nuevo sistema;
sudo chroot /mnt

Antes de instalar el grub hay que “crear” el *shell* sh emulándolo con bash, ya que lo necesita:
ln /home/pfc/bin/bash /home/pfc/bin/sh;

Instalamos el grub:
grub-install --recheck /dev/sdb

Ahora hay que generar el fichero que se usará para iniciar(antes hay que copiar el kernel de /boot al directorio boot del pendrive, que tendrá un nombre parecido a vmlinuz-... y también un fichero que se llama initrd...):
grub-mkconfig > /boot/grub/grub.cfg;
Generating grub.cfg ...
Found linux image: /boot/vmlinuz-2.6.38-8-generic
Found initrd image: /boot/initrd.img-2.6.38-8-generic
done

Y por último para que el nuevo sistema se pueda iniciar, hay que indicarle cual será el proceso inicial. Para eso creamos un script /sbin/init **con permisos de ejecución** (755) que ejecute /bin/bash:
cat >/sbin/init;
#!/bin/bash

/bin/bash

Para que no haya problemas de usuarios, ponemos que el dueño de todos lo archivos sea root (ya desde fuera del nuevo sistema):
sudo chown -R root:root /media/PFC/*;

Ahora reiniciamos la máquina y en la BIOS ponemos el pendrive como primera opción, de esta manera arrancará desde el usb.

Pasamos a hacer la segunda versión.

### 7.2.3. Versión 2

Programas necesarios

- GNU:
    - E2fsprogs
    - Make
    - Gawk
    - Autoconf
    - Automake
    - Libtool
    - XFree86
    - Zlib
    - Perl
    - Libpng
    - FluxBox
    - Dillo
    - Fltk
    - Wget

    - Nagios
    - GD
    - Apache
    - libXML2

- No-GNU:
    - SysVinit
    - LFS-Bootscripts
    - Sysklogd
    - Udev
    - Module-init-tools
    - Libjpeg
    - PHP

- Análisis de estado:
    - Dillo
    - Fltk
    - Nagios
    - Libjpeg
    - GD
    - Apache
    - PHP
    - libXML2

SysVinit

Descarga:
sysVinit-2.88: http://download.savannah.gnu.org/releases/sysvinit/

Descomprimir:
~/Descargas$ tar xvfj sysvinit-2.88dsf.tar.bz2

Ya esta el Makefile hecho así que compilamos:
~/Descargas/sysvinit-2.88dsf$ make

Instalación:
~/Descargas/sysvinit-2.88dsf$ sudo make ROOT=/home/pfc install

Ahora hay que configurarlo. Para ello hay que crear el archivo /home/pfc/etc/inittab con el contenido adecuado:
sudo cat > /home/pfc/etc/inittab
# Begin /etc/inittab

id:3:initdefault:

si::sysinit:/etc/rc.d/init.d/rc sysinit

l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S016:once:/sbin/sulogin

1:2345:respawn:/sbin/agetty tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# End /etc/inittab

Para que el arranque se lleve a cabo directamente, se necesitan algunos ficheros de configuración.

LFS-Bootscripts[4]

Descarga:
lfs-bootscripts-20090812: http://downloads.linuxfromscratch.org/

Descomprimir:
~/Descargas$ tar xvfj lfs-bootscripts-20090812.tar.bz2

Ya está el Makefile listo, así que compilamos:
~/Descargas/lfs-bootscripts-20090812$ sudo make

Instalación:
~/Descargas/lfs-bootscripts-20090812$ sudo make DESTDIR=/home/pfc install

Hay alguna incongruencia en algún archivo que hay que solucionar:
En el archivo /home/pfc/etc/rc.d/init.d/network:

- . /etc/sysconfig/network -------> #. /etc/sysconfig/network [línea 19]

En el archivo /home/pfc/etc/rc.d/init.d/localnet hay que hacer lo mismo:

- . /etc/sysconfig/network -------> #. /etc/sysconfig/network [línea 17]

En el fichero /home/pfc/etc/rc.d/init.d/localnet se indica el nombre que tendrá el host a partir de la variable “HOSTNAME”, pero está vacía, así que al final del fichero /home/pfc/etc/sysconfig/rc, que es incluido por el fichero localnet, añadimos:
HOSTNAME=PFC

Al iniciar, el nuevo host recibirá el nombre ‘PFC’.

Hay que copiar los archivos necesarios para la gestión de usuarios:
sudo cp /etc/passwd /home/pfc/etc/;
sudo cp /etc/shadow /home/pfc/etc/;

sudo cp /etc/group /home/pfc/etc/;
sudo cp /etc/gshadow /home/pfc/etc/;

Temporalmente le quitamos la contraseña al usuario root para poder loguearnos sin problemas. En el archivo passwd, la primera línea que será como:
root:x:0:0:root:/root:/bin/bash

Hay que borrar la x, que es lo que indica que tiene contraseña, y así no pedirá la contraseña al entrar al sistema. Por lo tanto queda así:
root::0:0:root:/root:/bin/bash

Sysklogd

Descarga:
sysklogd-1.5: http://www.infodrom.org/projects/sysklogd/download/

Descomprimir:
~/Descargas$ tar xvfz sysklogd-1.5.tar.gz

Ya viene con el Makefile directamente, así que compilamos:
~/Descargas/sysklogd-1.5$ make

Instalación:
~/Descargas/sysklogd-1.5$ sudo make prefix=/home/pfc install

Para que, tanto el comando syslogd, como el klogd guarden información útil, se necesita el fichero de configuración syslog.conf que indica, entre otras cosas, los ficheros donde guardar esa información:
cat >/home/pfc/etc/syslog.conf
# /etc/syslog.conf
# For info about the format of this file, see "man syslog.conf"
# and /usr/doc/sysklogd/README.linux.  Note the '-' prefixing some
# of these entries;  this omits syncing the file after every logging.

```
# In the event of a crash, some log information might be lost, so
# if this is a concern to you then you might want to remove the '-'.
# Be advised this will cause a performation loss if you're using
# programs that do heavy logging.

# Uncomment this to see kernel messages on the console.
#kern.*                                         /dev/console

# Log anything 'info' or higher, but lower than 'warn'.
# Exclude authpriv, cron, mail, and news.  These are logged elsewhere.
*.info;*.!warn;\
    authpriv.none;cron.none;mail.none;news.none    -/var/log/messages

# Log anything 'warn' or higher.
# Exclude authpriv, cron, mail, and news.  These are logged elsewhere.
*.warn;\
    authpriv.none;cron.none;mail.none;news.none    -/var/log/syslog

# Debugging information is logged here.
#*.=debug                                       -/var/log/debug

# Private authentication message logging:
authpriv.*                                      -/var/log/secure

# Cron related logs:
cron.*                                          -/var/log/cron

# Mail related logs:
mail.*                                          -/var/log/maillog

# Emergency level messages go to all users:
*.emerg                                         *

# This log is for news and uucp errors:
```

uucp,news.crit -/var/log/spooler

# Uncomment these if you'd like INN to keep logs on everything.
# You won't need this if you don't run INN (the InterNetNews daemon).
#news.=crit -/var/log/news/news.crit
#news.=err -/var/log/news/news.err
#news.notice -/var/log/news/news.notice

Udev

Descarga:
udev-173: http://packages.ubuntu.com/source/oneiric/udev

Hay que crear unos directorios que necesita pero que no los crea por si mismo:
mkdir /home/pfc/lib/firmware
mkdir /home/pfc/lib/udev
mkdir /home/pfc/lib/udev/devices
mkdir /home/pfc/lib/udev/devices/pts
mknod -m0666 /home/pfc/lib/udev/devices/null c 1 3

Descomprimir:
~/Descargas$ tar xvfz udev_173.orig.tar.gz

Configuración:
~/Descargas/udev-173$ ./configure --prefix=/usr --disable-hwdb --disable-introspection --disable-keymap --disable-gudev

Compilación:
~/Descargas/udev-173$ make

Instalación:
~/Descargas/udev-173$ sudo make DESTDIR=/home/pfc install

Por último hacemos unos symbolic link en /home/pfc/sbin/, que es donde se buscarán los comandos al iniciar el sistema:

~/sbin$ ln -s ../usr/sbin/udevadm ./;

~/sbin$ ln -s ../usr/sbin/udevd ./;

En esta versión se monta el sistema de ficheros entero, que es ext4, por lo que al montarlo un número determinado de veces, se hará un chequeo del sistema de ficheros para ver que todo esté bien. Esto es un problema, ya que solo disponemos del comando fsck para el sistema de ficheros minix. La única solución es instalar ese comando para más sistemas de ficheros:

E2fsprogs

Descarga:

e2fsprogs-1.42: http://sourceforge.net/projects/e2fsprogs/

Descomprimir:

~/Descargas$ tar xvfz e2fsprogs-1.42.tar.gz

Configuración:

~/Descargas/e2fsprogs-1.42$ CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/lib ./configure --prefix=/usr

Compilación:

~/Descargas/e2fsprogs-1.42$ make

Instalación:

~/Descargas/e2fsprogs-1.42$ sudo make DESTDIR=/home/pfc install

Make

Descarga:

make-3.82: http://ftp.gnu.org/gnu/make/

Descomprimir:
~/Descargas$ tar xvfj make-3.82.tar.bz2

Configuración:
~/Descargas/make-3.82$ CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/lib ./configure --prefix=/usr

Compilación:
~/Descargas/make-3.82$ make

Instalación:
~/Descargas/make-3.82$ sudo make DESTDIR=/home/pfc install

Gawk

Descarga:
gawk-4.0.0: http://ftp.gnu.org/gnu/gawk/

Descomprimir:
~/Descargas$ tar xvfj gawk-4.0.0.tar.bz2

Configuración:
~/Descargas/gawk-4.0.0$ CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/lib ./configure --prefix=/usr

Compilación:
~/Descargas/gawk-4.0.0$ make

Instalación:
~/Descargas/gawk-4.0.0$ sudo make DESTDIR=/home/pfc install

Autoconf

Descarga:
autoconf-2.68: http://ftp.gnu.org/gnu/autoconf/

Descomprimir:
~/Descargas$ tar xvfj autoconf-2.68.tar.bz2

Configuración:
~/Descargas/autoconf-2.68$ CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/lib ./configure --prefix=/usr

Compilación:
~/Descargas/autoconf-2.68$ make

Instalación:
~/Descargas/autoconf-2.68$ sudo make DESTDIR=/home/pfc install

Automake

Descarga:
automake-1.11.2: http://ftp.gnu.org/gnu/automake/

Descomprimir:
~/Descargas$ tar xvfj automake-1.11.2.tar.bz2

Configuración:
~/Descargas/automake-1.11.2$ CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/lib ./configure --prefix=/usr

Compilación:
~/Descargas/automake-1.11.2$ make

Instalación:
~/Descargas/automake-1.11.2$ sudo make DESTDIR=/home/pfc install

Libtool

Descarga:
libtool-2.4.2: ftp://ftp.gnu.org/gnu/libtool/

Descomprimir:
~/Descargas$ tar xvfz libtool-2.4.2.tar.gz

Configuración:
~/Descargas/libtool-2.4.2$ CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/lib ./configure --prefix=/usr

Compilación:
~/Descargas/libtool-2.4.2$ make

Instalación:
~/Descargas/libtool-2.4.2$ sudo make DESTDIR=/home/pfc install

Module-init-tools

Descarga:
module-init-tools-3.15: http://ftp.kernel.org/pub/linux/utils/kernel/module-init-tools/

Descomprimir:
~/Descargas$ tar xvfj module-init-tools-3.15.tar.bz2

Configuración:
~/Descargas/module-init-tools-3.15$ ./configure --prefix=/usr

Antes de compilar, hay que comentar unas líneas del Makefile donde lo que hace es utilizar el comando “docbook2man” que no tenemos para crear unas páginas del manual, y así ahorrarnos el problema de instalarlo solo para eso. Las líneas son de la 1000 a la 1004 y de la 1007 a la 1013 que son en las que aparece este comando, y hacemos también que no instale los manuales borrando todo lo que contienen los apartados “install-

man5:" y "install-man8:".

Compilación:
~/Descargas/module-init-tools-3.15$ make

Instalación:
~/Descargas/module-init-tools-3.15$ sudo make DESTDIR=/home/pfc install

//////////////////////////////////////////////////backup 23-11-2011//////////////////////////////////////////

X-Window[5][6]

Dependencias: Zlib, Perl, Libpng

Descarga[7]:
XFree86-4.8.0: http://www.xfree86.org/

Descomprimir:
~/Descargas$ tar xvfz XFree86-4.8.0-src-1.tgz
~/Descargas$ tar xvfz XFree86-4.8.0-src-2.tgz
~/Descargas$ tar xvfz XFree86-4.8.0-src-3.tgz
~/Descargas$ tar xvfz XFree86-4.8.0-src-4.tgz
~/Descargas$ tar xvfz XFree86-4.8.0-src-5.tgz
~/Descargas$ tar xvfz XFree86-4.8.0-src-6.tgz
~/Descargas$ tar xvfz XFree86-4.8.0-src-7.tgz

Ahora hay que crear un directorio en el mismo nivel que el directorio descomprimido (~/Descargas) para compilar el paquete, y hacer symbolic links para poder compilarlo. Ésto es útil porque así mantenemos el paquete original sin modificar, y podemos guardar diferentes configuraciones en sus respectivos directorios:
~/Descargas$ mkdir xfree86_build;
~/Descargas/xfree86_build$ lndir ../xc;

Antes de compilar hay que solucionar algún problema que hay.

El primero es que buscará las bibliotecas limits.h en un lugar donde no están, las busca en ~/usr/lib/gcc/i686-pc-linux-gnu/4.6.1/include pero están en ~/usr/lib/gcc/i686-pc-linux-gnu/4.6.1/include-fixed, así que hacemos un symbolic link para arreglarlo:
~/usr/lib/gcc/i686-pc-linux-gnu/4.6.1/include$ ln -s ../include-fixed/limits.h ./

Al compilar necesita el fichero version.h, que está en el Ubuntu, pero no en el nuevo sistema, por lo tanto lo copiamos al nuevo sistema:
~/usr/include/linux$ cp /usr/include/linux/version.h ./

El archivo fbdevhw.c necesita la variable PAGE_MASK, que está en el archivo /home/pfc/usr/include/asm/page_types.h, que a su vez está incluido en el archivo /home/pfc/usr/include/asm/page.h, que es el que incluye el mencionado archivo fbdevhw.c; pero no tiene esa variable declarada, porque la estructura de esos ficheros se ha cambiado, y actualmente el fichero page.h solo incluye page_types.h si no están definidas las cabeceras del kernel.

Para solucionarlo, simplemente hay que añadir en el fichero /home/pfc/Descargas/xfree86_build/programs/Xserver/hw/xfree86/fbdevhw/fbdevhw.c la línea:
#include "asm/page_types.h"
justo debajo de:
#include "asm/page.h"  /* #define for PAGE_* */

Hay un bug del paquete, para arreglarlo hay que abrir el archivo /home/pfc/Descargas/xfree86_build/programs/Xserver/hw/tinyx/vesa/vm86.h, y debajo de:
#include “os.h”
#endif

añadir unas declaraciones que están en otro archivo:
#ifndef IF_MASK
#define IF_MASK X86_EFLAGS_IF
#endif
#ifndef IOPL_MASK

#define IOPL_MASK X86_EFLAGS_IOPL
#endif

Desde la versión de libpng-1.5.0 el struct png_info (que es lo que retorna la función png_create_info_struct) ya no se puede acceder desde las aplicaciones. Tenemos que cambiar la línea 185 de /home/pfc/Descargas/xfree86_build/programs/xcursorgen/xcursorgen.c:
if (setjmp (png->jmpbuf))

por esta otra:
if (setjmp (png_jmpbuf(png)))

Tal como se explica en el fichero png.h

Para que encuentre las bibliotecas libpng, hay que abrir el archivo /home/pfc/Descargas/xfree86_build/config/util/Imakefile y en la línea siguiente donde pone:
all:: xmkmf mergelib $(CCMDEP_PROG) $(GCCMDEP_PROG) $(PROGRAMS)

añadir:
ln -s /home/pfc/usr/lib/libpng15.so.15 /home/pfc/Descargas/xfree86_build/exports/lib/

Es muy importante que los espacios que hay delante de “ln” sean de apretar una vez la tecla tabulador, si no, el make fallará.

Compilación:
~/Descargas/xfree86_build$ make World > World.log

Instalación:
~/Descargas/xfree86_build$ sudo make DESTDIR=/home/pfc install

También hay que crear el directorio /var/log:
mkdir /home/pfc/var/log;

Las bibliotecas de las X no estarán incluidas en el nuevo sistema, así que hacemos symbolic links:
~/usr/lib$ ln -s ../X11R6/lib/* ./

Al iniciar al sistema...
Para configurar las X de manera automática hay que hacer:
XFree86 -configure

Más adelante se explica detalladamente cómo hacerlo.

Zlib

Descarga:
zlib_1.2.3.4: http://packages.ubuntu.com/source/oneiric/zlib

Descomprimir:
~/Descargas$ tar xvfz zlib_1.2.3.4.dfsg.orig.tar.gz

Configuración:
~/Descargas/zlib_1.2.3.4.dfsg$ ./configure --prefix=/usr

Compilación:
~/Descargas/zlib_1.2.3.4.dfsg$ make

Instalación:
~/Descargas/zlib_1.2.3.4.dfsg$ sudo make DESTDIR=/home/pfc install

Libpng

Descarga:
libpng-1.5.6: http://libpng.sourceforge.net/index.html

Descomprimir:

~/Descargas$ tar xvfj libpng-1.5.6.tar.bz2

Configuración:
~/Descargas/libpng-1.5.6$ ./configure --prefix=/usr

Compilación:
~/Descargas/libpng-1.5.6$ make

Instalación:
~/Descargas/libpng-1.5.6$ sudo make DESTDIR=/home/pfc install

Perl

Existe un bug al compilar perl en Ubuntu11.04, que es el que se está utilizando, así que hay que usar una utilidad que nos facilite la instalación:
curl -L http://xrl.us/perlbrewinstall | bash;
/home/pfc/perl5/perlbrew/bin/perlbrew install perl-5.12.3 -Dperllibs='-lm -lc';

Ahora abrimos el fichero ~/perl5/perlbrew/build/perl-5.12.3/config.sh y sustituimos todo lo que sea “/home/pfc/perl5/perlbrew/perls/perl-5.12.3” por “/usr”, y ejecutamos:
~/perl5/perlbrew/build/perl-5.12.3$ make;
~/perl5/perlbrew/build/perl-5.12.3$ sudo DESTDIR=/home/pfc make install;

FluxBox[8]

Descarga:
fluxbox-1.3.2: http://fluxbox.org/download/

Descomprimir:
~/Descargas$ tar xvfj fluxbox-1.3.2.tar.bz2

Configuración:
~/Descargas/fluxbox-1.3.2$ ./configure CPPFLAGS=-I/home/pfc/usr/X11R6/include LDFLAGS=-L/home/pfc/usr/X11R6/lib --prefix=/usr

Compilación:
~/Descargas/fluxbox-1.3.2$ make

Instalación:
~/Descargas/fluxbox-1.3.2$ sudo make DESTDIR=/home/pfc install

Para ejecutar el entorno gráfico:
XFree86 -configure;
XFree86 -xf86config [archivo_creado_por_el_anterior_comando] &;
export DISPLAY=:0.0;
startfluxbox &;

//////////////////////////////////////////////////backup_19/12/2011//////////////////////////////////////////////

Dillo

Dependencias: FLTK

Descarga:
dillo-3.0.2: http://www.dillo.org/download/

Descomprimir:
~/Descargas$ tar xvfj dillo-3.0.2.tar.bz2

Configuración:
~/Descargas/dillo-3.0.2$ ./configure CPPFLAGS=-I/home/pfc/usr/X11R6/include LDFLAGS=-L/home/pfc/usr/X11R6/lib --prefix=/usr

Compilación:
~/Descargas/dillo-3.0.2$ make

Instalación:
~/Descargas/dillo-3.0.2$ sudo make DESTDIR=/home/pfc install

FLTK

Descarga:
fltk-1.3.0: http://www.fltk.org/software.php

Descomprimir:
~/Descargas$ tar xvfz fltk-1.3.0-source.tar.gz

Configuración:
~/Descargas/fltk-1.3.0$ ./configure CPPFLAGS=-I/home/pfc/usr/X11R6/include LDFLAGS=-L/home/pfc/usr/X11R6/lib --prefix=/usr

Compilación:
~/Descargas/fltk-1.3.0$ make

Instalación:
~/Descargas/fltk-1.3.0$ sudo make DESTDIR=/home/pfc install

Configuración de la conexión a internet desde casa:

Configurar el DNS:
cat > home/pfc/etc/resolv.conf
nameserver 192.168.1.1

Y una vez iniciado el nuevo sistema habrá que hacer:

desde una tty cualquiera levanto la interfície 0:
ifconfig eth0 up;

Seguramente la ip del router será 192.168.1.1, así que le pongo una ip que probablemente esté libre:
ifconfig eth0 192.168.1.120;

Miro si hay un default gateway ya puesto mediante el comando “route -n”, y si no hay, lo añadimos:
route add default gw 192.168.1.1 netmask 255.255.255.0

Y listo, ya se podría ejecutar el navegador Dillo y navegar por internet.

Con el navegador podemos buscar paquetes que necesitemos e instalarlos, pero para descargarlos nos hace falta el comando wget:

Wget

Descarga:
wget-1.13.4: http://ftp.gnu.org/gnu/wget/

Descomprimir:
~/Descargas$ tar xvfj wget-1.13.4.tar.bz2

Configuración:
~/Descargas/wget-1.13.4$ ./configure CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/lib --prefix=/usr --without-ssl

Compilación:
~/Descargas/wget-1.13.4$ make

Instalación:
~/Descargas/wget-1.13.4$ sudo make DESTDIR=/home/pfc install

Para la instalación de nagios nos hacen falta 2 paquetes principales (a parte de las dependencias): nagios3-core y nagios-plugins.

Nagios[9]

Dependencias: Libjpeg, GD, Apache, PHP

Descarga del primer paquete (nagios3-core):
nagios-3.3.1: http://www.nagios.org/download/core/thanks/

Descomprimir:
~/Descargas$ tar xvfz nagios-3.3.1.tar.gz

Configuración:
~/Descargas/nagios$ ./configure CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/lib --prefix=/usr

Creamos temporalmente un usuario nagios para la instalación:
sudo adduser nagios;

Compilación:
~/Descargas/nagios$ make all

Antes de instalar hay que hacer un symbolic link del archivo gd.h de el nuevo sistema al Ubuntu para que lo encuentre al instalar:
/usr/include$ sudo ln -s /home/pfc/usr/include/gd* ./

Hay un error en el Makefile, donde al instalar lo que hay en un directorio, dentro hay subdirectorios y esto hará que no pueda instalar. Para omitirlos hay que cambiar:

```
for file in includes/rss/*; \
   do $(INSTALL) -m 664 $(INSTALL_OPTS) $$file $(DESTDIR)$(HTMLDIR)/includes/rss; done
```

por:

```
for file in includes/rss/c*; \
   do $(INSTALL) -m 664 $(INSTALL_OPTS) $$file $(DESTDIR)$(HTMLDIR)/includes/rss; done
  for file in includes/rss/r*; \
```

do $(INSTALL) -m 664 $(INSTALL_OPTS) $$file $(DESTDIR)$(HTMLDIR)/includes/rss; done

También hay que crear unos directorios para la configuración:
~$ mkdir etc/httpd
~$ mkdir etc/httpd/conf.d

Instalación:
~/Descargas/nagios$ sudo make DESTDIR=/home/pfc install
~/Descargas/nagios$ sudo make DESTDIR=/home/pfc install-init
~/Descargas/nagios$ sudo make DESTDIR=/home/pfc install-config
~/Descargas/nagios$ sudo make DESTDIR=/home/pfc install-commandmode
~/Descargas/nagios$ sudo make DESTDIR=/home/pfc install-webconf

Ahora creamos el usuario nagiosadmin para poder acceder:
sudo htpasswd -c /home/pfc/usr/etc/htpasswd.users nagiosadmin
Y de contraseña ponemos también nagiosadmin.

Descarga del siguiente paquete (nagios-plugins):
nagios-plugins-1.4.15: http://www.nagios.org/download/plugins/

Descomprimir:
~/Descargas$ tar xvfz nagios-plugins-1.4.15.tar.gz

Configuración:
~/Descargas/nagios-plugins-1.4.15$ ./configure CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/lib --with-nagios-user=nagios --with-nagios-group=nagios --prefix=/usr

Compilación:
~/Descargas/nagios-plugins-1.4.15$ make

Instalación:

~/Descargas/nagios-plugins-1.4.15$ sudo make DESTDIR=/home/pfc install

Libjpeg

Descarga:
libjpeg-8c: http://www.ijg.org/

Descomprimir:
~/Descargas$ tar xvfz jpegsrc.v8c.tar.gz

Configuración:
~/Descargas/jpeg-8c$ ./configure CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/lib --prefix=/usr

Configuración:
~/Descargas/jpeg-8c$ make

Instalación:
~/Descargas/jpeg-8c$ sudo make DESTDIR=/home/pfc install

GD

Descarga:
gd-2.0.33: https://bitbucket.org/pierrejoye/gd-libgd/downloads

Descomprimir:
~/Descargas$ tar xvfj pierrejoye-gd-libgd-5551f61978e3.tar.bz2

Configuración:
~/Descargas/pierrejoye-gd-libgd-5551f61978e3/src$ ./configure CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/lib --prefix=/usr --with-gd-lib --with-gd-inc

Compilación:

~/Descargas/pierrejoye-gd-libgd-5551f61978e3/src$ make

Instalación:
~/Descargas/pierrejoye-gd-libgd-5551f61978e3/src$ sudo make DESTDIR=/home/pfc install

Apache

Descarga:
httpd-2.2.21: http://httpd.apache.org/download.cgi

Descomprimir:
~/Descargas$ tar xvfj httpd-2.2.21.tar.bz2

Configuración:
~/Descargas/httpd-2.2.21$ ./configure CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/lib --prefix=/usr --enable-so

Compilación:
~/Descargas/httpd-2.2.21$ make

Instalación:
~/Descargas/httpd-2.2.21$ sudo make DESTDIR=/home/pfc install

Después de instalar apache hay que revisar el fichero de configuración /home/pfc/usr/conf/httpf.conf, porque como no existe el usuario “web”, pondrá seguramente el primero de la lista que no sea root (en mi caso daemon). Si pasa esto, es probable que ese usuario no tenga permiso para acceder al servidor web. Como hemos copiado el fichero passwd del sistema original, podemos utilizar el usuario “pfc”, así que abrimos el fichero de configuración y donde pone User y Group, cambiamos el que haya puesto por “pfc”.

PHP[10]

Por opciones de la instalación, hay que instalarlo desde el nuevo sistema, haciendo los mount en /mnt y luego el chroot; copiamos el paquete en, por ejemplo, /home/pfc/root, y hacemos lo siguiente:

Dependencias: libXML2

Descarga:
php-5.3.8: http://www.php.net/downloads.php

Descomprimir:
/root$ tar xvfj php-5.3.8.tar.bz2

Antes de configurar hay que arreglar un bug en el archivo configure. Hay que sustituir:

```
if { (eval echo configure:21707: \"$ac_link\") 1>&5; (eval $ac_link) 2>&5; } && test -s conftest${ac_exeext} && (./conftest; exit) 2>/dev/null
then
        LIBS=$old_LIBS

        php_cv_libxml_build_works=yes

else
 echo "configure: failed program was:" >&5
 cat conftest.$ac_ext >&5
 rm -fr conftest*
        LIBS=$old_LIBS

        echo "$ac_t""no" 1>&6
        { echo "configure: error: build test failed.  Please check the config.log for details." 1>&2; exit 1; }
fi
```

Por (eliminar la comparación que hace, que da un error):

```
LIBS=$old_LIBS
```

php_cv_libxml_build_works=yes

Configuración:
/root/php-5.3.8$ CPPFLAGS=-I/usr/include LDFLAGS=-L/usr/lib ./configure --prefix=/usr --with-apxs2=/usr/bin/apxs

Para que se compile sin problemas hay que renombrar la libz, ya que la busca con otro nombre:
ln -s /usr/lib/libz.so /usr/lib/libz.so.1

Compilación:
/root/php-5.3.8$ make

Al instalar usa el comando sed, pero lo buscará en /bin, así que hacemos un symbolic link:
ln -s /usr/bin/sed /bin/

Por último hay que comentar una línea del Makefile donde intenta instalar un archivo que se ha omitido a la hora de compilar. Comentamos la línea “$(INSTALL) ext/phar/phar.phar $(INSTALL_ROOT)$(bindir)” que está en el apartado “install-pharcmd” del Makefile.

Instalación:
/root/php-5.3.8$ make install

Copiamos el fichero de configuración necesario para su funcionamiento:
/root/php-5.3.8$ cp php.ini-development /usr/lib/php/php.ini

Para que apache reconozca php hay que indicárselo, en el archivo /usr/conf/httpd.conf hay que añadir en el apartado de LoadModule (línea 53):
LoadModule php5_module modules/libphp5.so

Y en la sección de AddType añadimos:
AddType application/x-httpd-php .php .phtml
AddType application/x-httpd-php-source.phps

También hay que poner en el apartado correspondiente de ese mismo archivo como usuario y grupo al usuario “pfc”.

Ya está todo hecho, así que borramos el directorio de php para que no ocupe espacio en el sistema (/home/pfc/root/php-5.3.8)

LibXML2

Descarga:
libxml2-2.7.8: ftp://xmlsoft.org/libxml2/

Descomprimir:
~/Descargas$ tar xvfz libxml2-2.7.8.tar.gz

Configuración:
~/Descargas/libxml2-2.7.8$ ./configure CPPFLAGS=-I/home/pfc/usr/include LDFLAGS=-L/home/pfc/usr/lib --prefix=/usr

Compilación:
~/Descargas/libxml2-2.7.8$ make

Instalación:
~/Descargas/libxml2-2.7.8$ sudo make DESTDIR=/home/pfc install

Para que Apache encuentre a Nagios hay que indicarle donde está su archivo de configuración, para eso abrimos el fichero /home/pfc/usr/conf/httpd.conf y al final, antes de:
<IfModule ssl_module>
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
</IfModule>

añadimos:

Include /etc/httpd/conf.d

Y donde pone:
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>

Lo cambiamos por:
<IfModule dir_module>
    DirectoryIndex index.html index.php
</IfModule>

Ponemos como usuario y grupo a “pfc” dentro del archivo /home/pfc/usr/etc/nagios.cfg y del archivo /home/pfc/etc/rc.d/init.d/nagios.

Comprobamos que todo esté bien(esto solo funciona si se hace una vez iniciado el nuevo sistema):
/usr/bin/nagios -v /usr/etc/nagios.cfg;

Para configurarlo primero hay que crear el directorio /home/pfc/usr/etc/conf.d:
mkdir /home/pfc/usr/etc/conf.d

Ahora creamos unos archivos de configuración más intuitivos que nos ayudarán a configurar más fácilmente el programa(para ver el contenido de los ficheros ver el AnexoB->Nagios):

- /home/pfc/usr/etc/conf.d/hosts.cfg

- home/pfc/usr/etc/conf.d/hostgroups.cfg: en este fichero se definen grupos de máquinas los cuales uno quiere monitorizar algo en especial. Por ejemplo, si se quiere monitorizar el servidor web, se crea un grupo con los nombres de las máquinas separados por comas que tienen un servidor web, y al crear los servicios de monitorización, se le indica el grupo correspondiente.

- home/pfc/usr/etc/conf.d/services.cfg: en este fichero se ponen los servicios que se van a monitorizar y a que máquinas se le va a hacer dicha monitorización. La monitorización se hace mediante comandos de Nagios. En el fichero /usr/etc/obj/command.cfg hay algún ejemplo específico, y mediante los comandos que hay en el directorio /usr/libexec que se llaman check_... uno mismo puede hacer los que quiera que haya disponibles. Como se ve a continuación, los parámetros que necesitan los comandos se les pasan separados por exclamaciones (“!”).

- home/pfc/usr/etc/conf.d/contacts.cfg:

- home/pfc/usr/etc/conf.d/generic-host.cfg

- home/pfc/usr/etc/conf.d/generic-service.cfg

- home/pfc/usr/etc/cgi.cfg: para que la web se refresque cada 5 segundos hay que modificar:
  refresh_rate=5

Ahora hay que modificar el archivo /home/pfc/usr/etc/nagios.cfg para indicarle los nuevos archivos de configuración. En la parte donde indica los ficheros de configuración (“cfg_file=...”), ponemos todos los archivos de configuración que utiliza nagios, por lo tanto, son los que hay que modificar para cambiar cualquier cosa siguiendo las pautas que he utilizado al crearlos:
cfg_file=/usr/etc/objects/commands.cfg
cfg_file=/usr/etc/objects/timeperiods.cfg
cfg_file=/usr/etc/conf.d/hosts.cfg
cfg_file=/usr/etc/conf.d/services.cfg
cfg_file=/usr/etc/conf.d/hostgroups.cfg
cfg_file=/usr/etc/conf.d/contacts.cfg
cfg_file=/usr/etc/conf.d/generic-service.cfg
cfg_file=/usr/etc/conf.d/generic-host.cfg

El programa Nagios busca el comando ping en /bin, así que hay que hacer un symbolic

link:
~/bin$ ln -s ../usr/bin/ping ./

Ahora pasamos a preparar el sistema para que cuando se inicie arranque los servicios necesarios y haga las configuraciones apropiadas:

Preparamos la red para que se autoconfigure. Para ello hay que modificar el script /home/pfc/etc/rc.d/rc3.d/S20network:

en el start ponemos:
ifconfig eth0 up;
sleep 2;
ifconfig eth0 192.168.1.120;
route add default gateway 192.168.1.1;

En el arranque del sistema se busca el comando modprobe en /sbin, hacemos un symbolic link:
~/sbin$ ln -s ../usr/sbin/modprobe ./

En el archivo /home/pfc/etc/rc.d/init.d/cleanfs, que se ejecuta al iniciar el sistema, intenta borrar lo que hay en unos directorios que no hemos creado en la estructura de ficheros, así que los creamos. Ésta es la parte donde intenta borrar:

```
boot_mesg -n " /var/lock" ${NORMAL}
        cd /var/lock &&
        find . -type f -exec rm -f {} \; || failed=1

        boot_mesg " /var/run" ${NORMAL}
        cd /var/run &&
        find . ! -type d ! -name utmp \
                -exec rm -f {} \; || failed=1
        > /var/run/utmp
        if grep -q '^utmp:' /etc/group ; then
chmod 664 /var/run/utmp
chgrp utmp /var/run/utmp
```

```
fi
```

Así que hay que hacer:

```
mkdir /home/pfc/var/run;
mkdir /home/pfc/var/lock;
```

Ahora añadimos los servicios de Apache y Nagios al arranque del sistema:

```
~/etc/rc.d/rc3.d$ ln -s ../../../usr/bin/apachectl ./S30apache;
~/etc/rc.d/rc3.d$ ln -s ../init.d/nagios ./S40nagios;
```

Hay que crear el directorio /dev/pts para poder ejecutar terminales en el entorno gráfico:

```
mkdir /home/pfc/dev/pts;
```

Creamos el archivo fstab para tener todos los sistemas de archivos correctamente:

```
cat > /home/pfc/etc/fstab
#<file system> <mount point>   <type>          <options>              <dump> <pass>
/dev/sdb1      /               ext4            defaults                    0 1
none           /proc           proc            defaults                    0 0
none           /sys            sysfs           nodev,noexec,nosuid         0 0
none           /dev            devtmpfs,tmpfs  mode=0755                   0 0
none           /dev/pts        devpts          noexec,nosuid,gid=tty,mode=0620   0 0
```

Y por último, creamos un nuevo servicio que lo único que haga es ejecutar lo que falta (poner el teclado en español, ejecutar el entorno gráfico):

```
cat >/home/pfc/etc/rc.d/init.d/lasttools;
#!/bin/sh

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
  start)
        loadkeys es.map
        export PATH=/usr/X11R6/bin/:$PATH
```

```
        /usr/X11R6/XFree86 -xf86config /root/XFree86.new &
        export DISPLAY=:0.0
                sleep 4
        startfluxbox &
        ;;

  *)
        echo "Usage: ${0} {start}"
        exit 1
        ;;
esac

# End /etc/rc.d/init.d/lasttools
```

Este script habrá que ejecutarlo una vez que se haya iniciado la sesión como root, para simplificarlo hacemos un symbolic link desde el directorio /usr/bin para poder ejecutarlo directamente:
chmod 755 /home/pfc/etc/rc.d/init.d/lasttools;
/home/pfc/usr/bin$ ln -s ../../etc/rc.d/init.d/lasttools ./lasttools

Antes de poder ejecutarlo todo, hay que configurar las X para que se ejecuten sin errores, por lo tanto arrancamos la maquina con el pendrive (siguiendo los mismos pasos que al pasar la versión 1 al pendrive, pero sin crear el fichero /etc/init), una vez ha cargado nos logueamos como root. Ahora estaremos en /root, como no se ha ejecutado el último script que hemos hecho (lasttools), hay que hacer:
loadkeys es.map;
export PATH=/usr/X11R6/bin/:$PATH;

Ahora configuramos las X:
XFree86 -configure

Se crea el archivo /root/XF86Config.new cuyo contenido se puede ver en AnexoB->Servidor X->Por defecto.

Con este archivo ya se pueden iniciar las X, pero al poner el dispositivo del ratón le llama “Mouse0”, y no está definido con este nombre, así que hay que borrar cualquier línea que haga acepción al ratón. Al principio de todo:

```
InputDevice  "Mouse0" "CorePointer"
```

Y por la mitad del documento:

```
Section "InputDevice"
    Identifier  "Mouse0"
    Driver      "mouse"
    Option      "Protocol" "auto"
    Option      "Device" "/dev/mouse"
EndSection
```

Si ahora se iniciaran las X, la resolución sería de 640x480, la mínima. No es un problema grave, pero da una sensación extraña porque normalmente estamos acostumbrados (y también son mucho más cómodas) a resoluciones mucho más altas.

Para ver qué resoluciones hay disponibles hay que ejecutar:

```
XFree86 -verbose 6
```

En mi caso veo que las disponibles son 1280x800, 1024x768, 800x600 y 640x480. Para poder modificar esto hay que tener en cuenta unos parámetros que se llaman V-sync y H-sync, que son la frecuencia de sincronización de pantalla vertical y horizontal. Estos valores suelen venir especificados en los manuales de la pantalla. En mi caso son, respectivamente, 56.0 - 65.0 y 31.5-50.0. Para poder poner la resolución que se quiera, también hay que definir el modelo que se quiere, ya que a veces no lo reconoce[11]. Para esto existe el comando cvt, así que en mi Ubuntu ejecuto:

```
cvt 1280 800;
# 1280x800 59.81 Hz (CVT 1.02MA) hsync: 49.70 kHz; pclk: 83.50 MHz
Modeline "1280x800_60.00"   83.50  1280 1352 1480 1680  800 803 809 831 -hsync +vsync
```

La línea que importa es la segunda, pero al copiarla al archivo de configuración hay que

cambiar la ‘M’ mayúscula por minúscula. También cambio el nombre "1280x800_60.00" por "1280x800” por comodidad.

Ahora cambiamos lo necesario en el archivo de configuración de XFree.

En “Section “Monitor” ” hay que añadir al final:
HorizSync 31.5-50.0
VertRefresh 56.0 - 65.0
modeline "1280x800" 83.50 1280 1352 1480 1680 800 803 809 831 -hsync +vsync

Y en “Section “Screen” “, debajo de “Depth 16”:
Modes "1280x800"

El contenido final del fichero se puede ver en AnexoB->Servidor X->Corregido.

Reinicio con el comando:
shutdown -r now

Me logueo como root y ejecuto:
/root$ lasttools start

# 8. Pruebas

Para ir comprobando el buen funcionamiento tanto de la versión 1 como de la versión 2, se han utilizado los programas instalados para ver si funcionaban correctamente.

Mostraremos algunos ejemplos:

Ping

Hemos hecho la prueba básica de utilizarlo con la IP de la interfície loopback. También la de la propia máquina, con una IP de router, de la otra máquina que tengo en casa, de google, y cuando Nagios lo utiliza con todas las máquinas que le he puesto, ha funcionado correctamente.

Find y grep

Durante el desarrollo del proyecto, ha sido necesario buscar muchos archivos o palabras clave para hacer configuraciones y arreglar aspectos del nuevo sistema, por lo que ha sido necesario usar *find* y *grep*. También se ha comprobado que los resultados obtenidos en el nuevo sistema son los mismos que los obtenidos en el sistema Ubuntu local.

La red

Para comprobar el funcionamiento de la red también es muy útil el comando ping, ya que para que funcione de la manera que lo hace hace falta que la red funcione correctamente. También lo he comprobado más exhaustivamente utilizando el navegador instalado, *dillo*, con el cual he buscado soluciones a algún problema que tenía estando ya con la versión 2 en marcha.

Make + autoconf + automake

He configurado, compilado e instalado el paquete php desde dentro de la versión 2, y se ha instalado sin problemas por lo que estos tres comandos funcionan perfectamente.

Wget + make + autoconf + automake

Para poder hacer capturas de pantalla y poner ejemplos visuales, se ha instalado el paquete ImageMagick desde la versión 2. Este software se ha podido descargar sin problemas desde el navegador (que utiliza el comando wget) y también se ha podido instalar sin problemas.

Bzip2, gzip, tar

Se han utilizado los comandos para desempaquetar los paquetes php y ImageMagick para poder instalarlos.

Otros programas que se ha comprobado que funcionan: grub, cp, cd, mkdir, xterm, mount, ls, ifconfig, loadkeys, vim, xedit, el servidor web...

# 9. Evaluación

## 9.1. Arreglar grub

Para evaluar el correcto funcionamiento de la versión 1, he “estropeado” el fichero /boot/grub/grub.cfg del PC de sobremesa que tengo en casa eliminando todas las entradas que tengo al SO Ubuntu y a Windows, de manera que es imposible arrancar en ese sistema operativo.

Teniendo la versión 1 del proyecto en un pendrive, he iniciado la máquina con él conectado y en la BIOS he cambiado el orden de arranque para que el boot de la máquina se realice desde el pendrive.

Ha arrancado desde el pendrive, el sistema operativo (versión 1 de este proyecto) ha arrancado correctamente y ha presentado la tty1, lista para aceptar comandos.

A continuación, para poder arreglarlo he seguido estos pasos:

He utilizado el comando “fdisk -l” para saber la partición donde está instalado Ubuntu:
tcsh-4.2# fdisk -l

Disk /dev/sda: 40.0 GB, 40020664320 bytes
255 heads, 63 sectors/track, 4865 cylinders, total 78165360 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xac7aac7a

| Device | Boot | Start | End | Blocks | Id | System |
|---|---|---|---|---|---|---|
| /dev/sda1 | * | 63 | 16402364 | 8201151 | 7 | HPFS/NTFS/exFAT |

```
/dev/sda2     16402365    65143574    24370605     7  HPFS/NTFS/exFAT
/dev/sda3     65143575    78156224    6506325      7  HPFS/NTFS/exFAT

Disk /dev/sdb: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders, total 156301488 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x12e612e5

Device Boot          Start          End    Blocks   Id  System
/dev/sdb1   *          63   156280319     78140128+   7  HPFS/NTFS/exFAT

Disk /dev/sdc: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders, total 156301488 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x2885db22

   Device Boot          Start          End    Blocks   Id  System
/dev/sdc1   *          63      78156224     39078081     7  HPFS/NTFS/exFAT
/dev/sdc2      78156225   156296384     39070080   83  Linux

Disk /dev/sdd: 1010 MB, 1010827264 bytes
2 heads, 63 sectors/track, 15668 cylinders, total 1974272 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0192aaa4

   Device Boot          Start          End    Blocks   Id  System
/dev/sdd1   *          32      1974271      987120        b  W95 FAT32
```

Se ve que la partición es /dev/sdc2.

Seguidamente he montado la partición de una manera similar a como se hacía para la versión 1:

```
sudo mount /dev/sdc2 /mnt
sudo mount --bind /proc /mnt/proc
sudo mount --bind /dev /mnt/dev
sudo mount --bind /sys /mnt/sys
```

Y ahora accedo al sistema local;

```
sudo chroot /mnt
```

Reinstalo el grub:

```
grub-install --recheck /dev/sdc
```

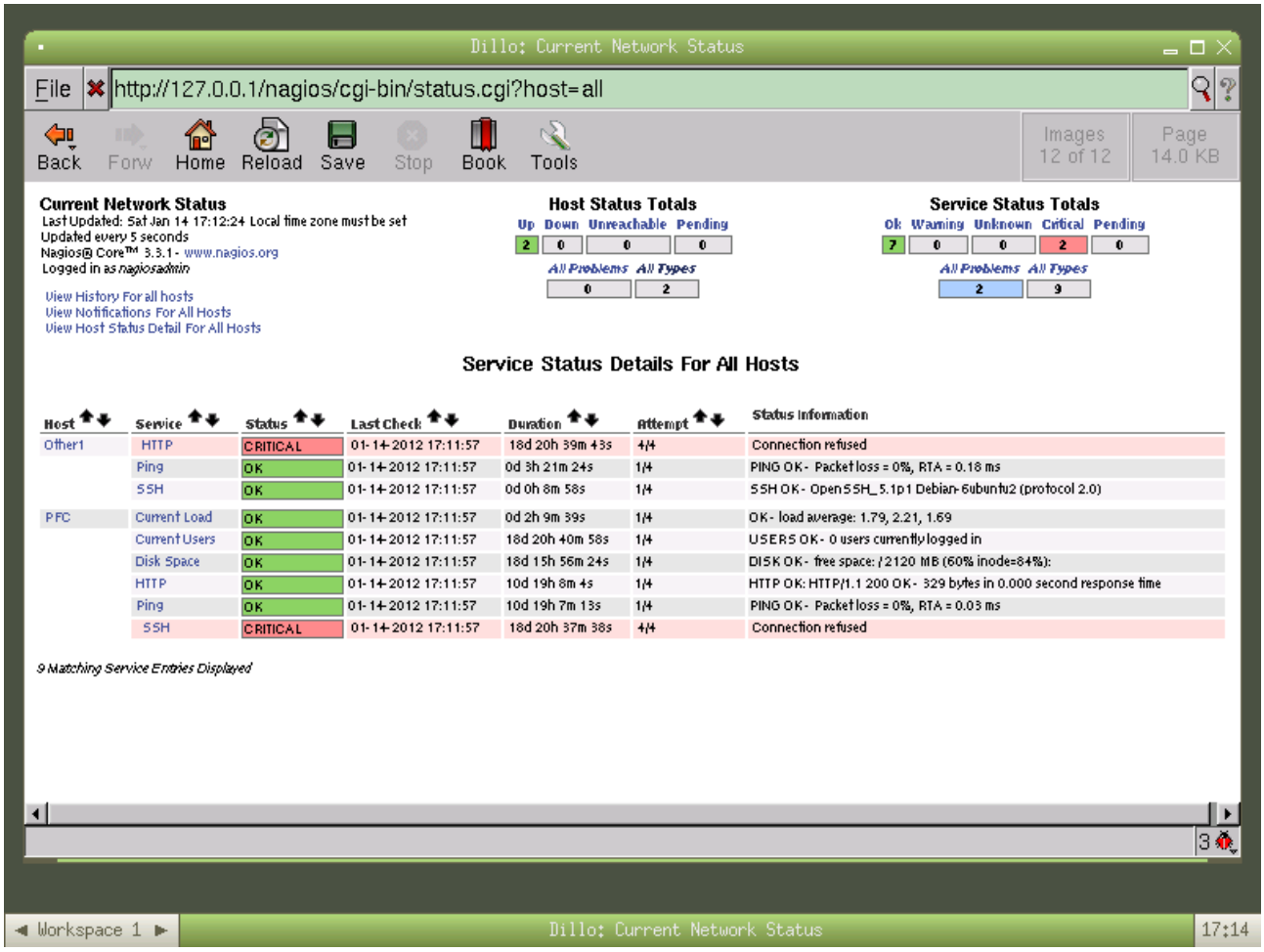
El problema que tengo ahora, es que al arrancar el PC solo me deja seleccionar el sistema Ubuntu. La entrada de Windows no se ha recuperado. Para arreglar esto último, extraigo el pendrive e inicio en el Ubuntu local, y en una terminal ejecuto:

```
sudo update-grub2
```

Ahora ya estará todo correcto y el ordenador podrá volver a arrancar desde el disco duro, tanto el sistema Linux Ubuntu, como Windows.

## *9.2. Determinar el estado de la red*

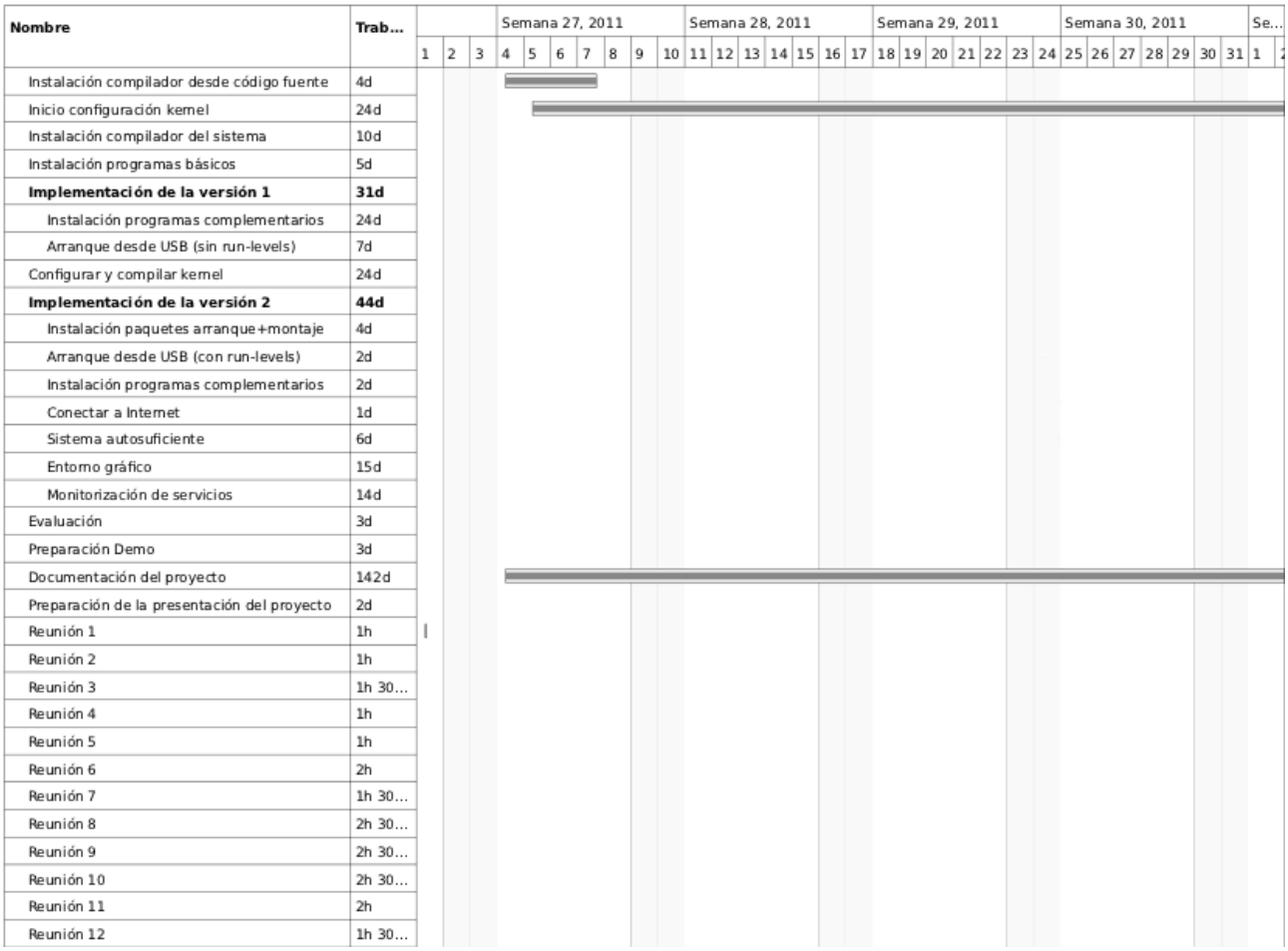
Para ver el funcionamiento de la versión 2 voy a monitorizar las dos máquinas que tengo en casa: el portátil, donde tendré corriendo la versión 2, y el de sobremesa, donde tendré un Ubuntu. Tal como tengo los archivos de configuración de Nagios (explicado en la parte de implementación), ya puedo monitorizar estas dos máquinas, y se obtiene este resultado:




Como se puede ver, lo único que falla es el servidor ssh de la versión 2 (PFC), ya que no lo tiene instalado, y el servidor web del PC de sobremesa (Other1), que tampoco lo tiene instalado, y, por lo tanto, la monitorización es correcta.

# 10. Planificación del proyecto

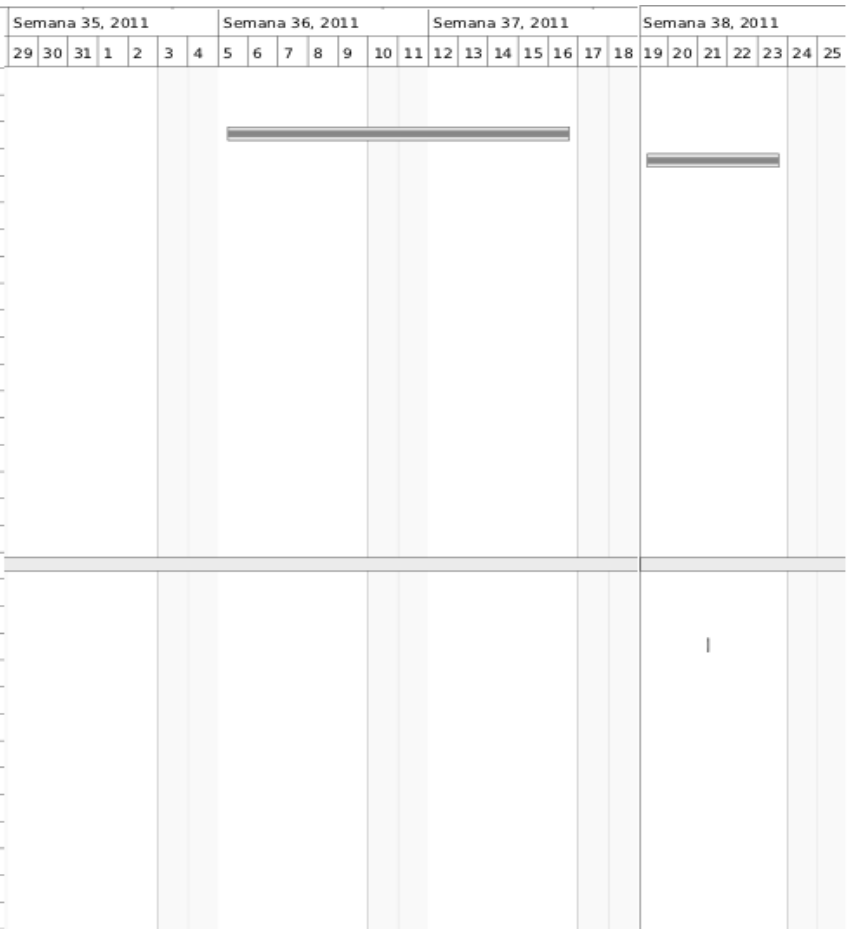
En esta sección se presenta el diagrama de Gantt del proyecto, con la tareas en las que se ha dividido el proyecto.

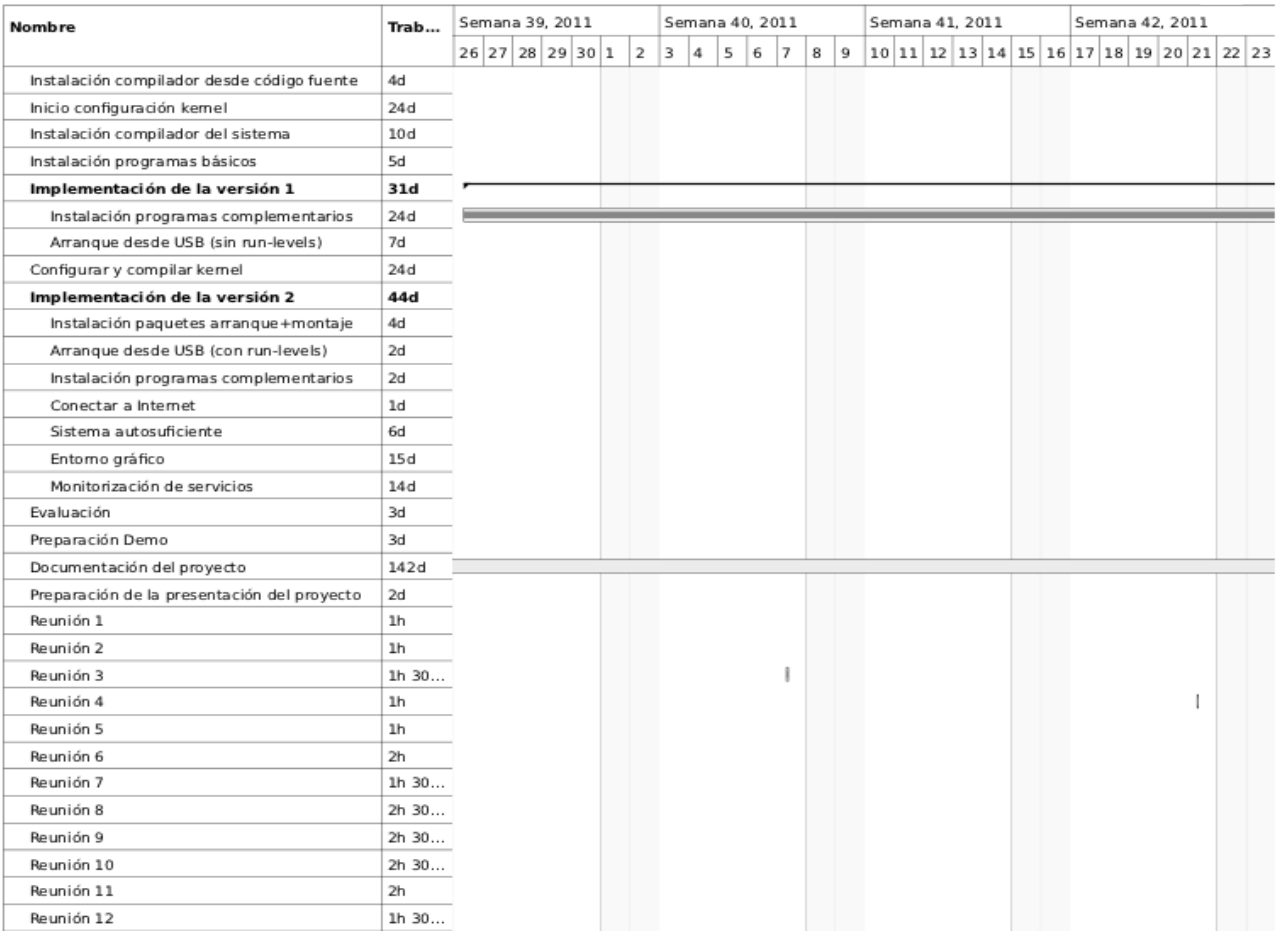


| Nombre | Trab... |
|---|---|
| Instalación compilador desde código fuente | 4d |
| Inicio configuración kernel | 24d |
| Instalación compilador del sistema | 10d |
| Instalación programas básicos | 5d |
| **Implementación de la versión 1** | **31d** |
| Instalación programas complementarios | 24d |
| Arranque desde USB (sin run-levels) | 7d |
| Configurar y compilar kernel | 24d |
| **Implementación de la versión 2** | **44d** |
| Instalación paquetes arranque+montaje | 4d |
| Arranque desde USB (con run-levels) | 2d |
| Instalación programas complementarios | 2d |
| Conectar a Internet | 1d |
| Sistema autosuficiente | 6d |
| Entorno gráfico | 15d |
| Monitorización de servicios | 14d |
| Evaluación | 3d |
| Preparación Demo | 3d |
| Documentación del proyecto | 142d |
| Preparación de la presentación del proyecto | 2d |
| Reunión 1 | 1h |
| Reunión 2 | 1h |
| Reunión 3 | 1h 30... |
| Reunión 4 | 1h |
| Reunión 5 | 1h |
| Reunión 6 | 2h |
| Reunión 7 | 1h 30... |
| Reunión 8 | 2h 30... |
| Reunión 9 | 2h 30... |
| Reunión 10 | 2h 30... |
| Reunión 11 | 2h |
| Reunión 12 | 1h 30... |

| Nombre | Trab... |
|---|---|
| Instalación compilador desde código fuente | 4d |
| Inicio configuración kernel | 24d |
| Instalación compilador del sistema | 10d |
| Instalación programas básicos | 5d |
| **Implementación de la versión 1** | **31d** |
| Instalación programas complementarios | 24d |
| Arranque desde USB (sin run-levels) | 7d |
| Configurar y compilar kernel | 24d |
| **Implementación de la versión 2** | **44d** |
| Instalación paquetes arranque+montaje | 4d |
| Arranque desde USB (con run-levels) | 2d |
| Instalación programas complementarios | 2d |
| Conectar a Internet | 1d |
| Sistema autosuficiente | 6d |
| Entorno gráfico | 15d |
| Monitorización de servicios | 14d |
| Evaluación | 3d |
| Preparación Demo | 3d |
| Documentación del proyecto | 142d |
| Preparación de la presentación del proyecto | 2d |
| Reunión 1 | 1h |
| Reunión 2 | 1h |
| Reunión 3 | 1h 30... |
| Reunión 4 | 1h |
| Reunión 5 | 1h |
| Reunión 6 | 2h |
| Reunión 7 | 1h 30... |
| Reunión 8 | 2h 30... |
| Reunión 9 | 2h 30... |
| Reunión 10 | 2h 30... |
| Reunión 11 | 2h |
| Reunión 12 | 1h 30... |

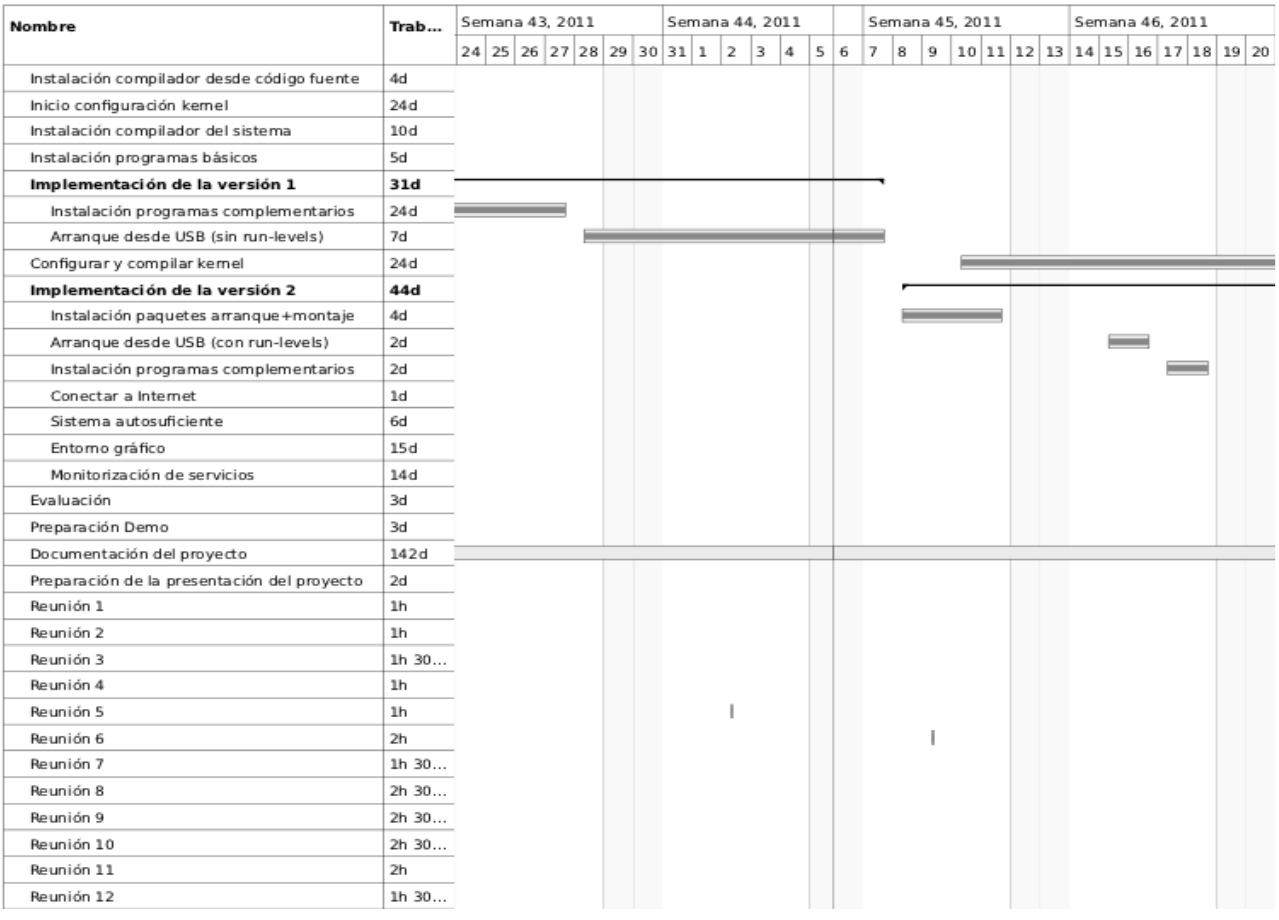

| Nombre |
|---|
| Instalación compilador desde código fuente |
| Inicio configuración kernel |
| Instalación compilador del sistema |
| Instalación programas básicos |
| **Implementación de la versión 1** |
| Instalación programas complementarios |
| Arranque desde USB (sin run-levels) |
| Configurar y compilar kernel |
| **Implementación de la versión 2** |
| Instalación paquetes arranque+montaje |
| Arranque desde USB (con run-levels) |
| Instalación programas complementarios |
| Conectar a Internet |
| Sistema autosuficiente |
| Entorno gráfico |
| Monitorización de servicios |
| Evaluación |
| Preparación Demo |
| Documentación del proyecto |
| Preparación de la presentación del proyecto |
| Reunión 1 |
| Reunión 2 |
| Reunión 3 |
| Reunión 4 |
| Reunión 5 |
| Reunión 6 |
| Reunión 7 |
| Reunión 8 |
| Reunión 9 |
| Reunión 10 |
| Reunión 11 |
| Reunión 12 |



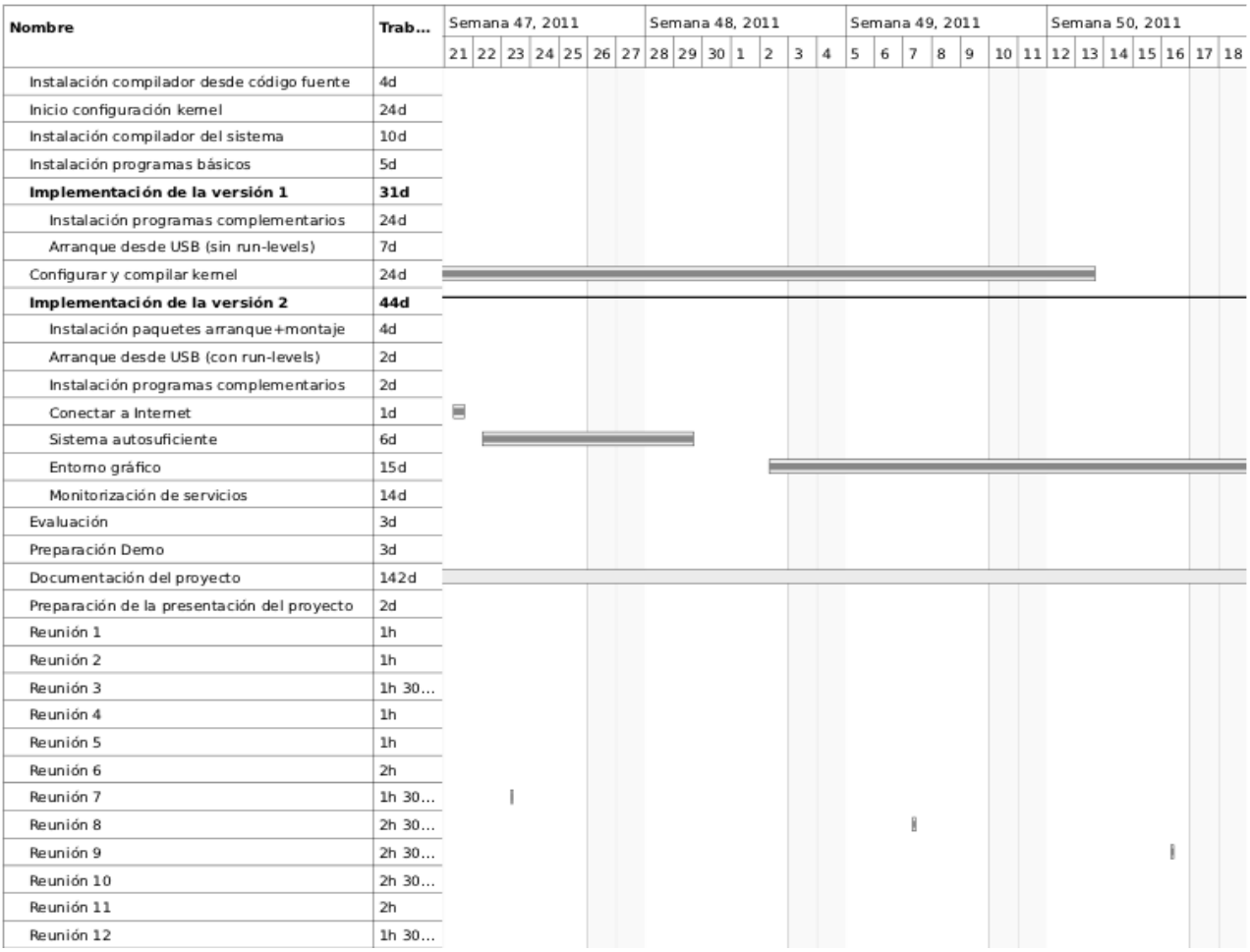


| Nombre | Trab... |
|---|---|
| Instalación compilador desde código fuente | 4d |
| Inicio configuración kernel | 24d |
| Instalación compilador del sistema | 10d |
| Instalación programas básicos | 5d |
| **Implementación de la versión 1** | **31d** |
| Instalación programas complementarios | 24d |
| Arranque desde USB (sin run-levels) | 7d |
| Configurar y compilar kernel | 24d |
| **Implementación de la versión 2** | **44d** |
| Instalación paquetes arranque+montaje | 4d |
| Arranque desde USB (con run-levels) | 2d |
| Instalación programas complementarios | 2d |
| Conectar a Internet | 1d |
| Sistema autosuficiente | 6d |
| Entorno gráfico | 15d |
| Monitorización de servicios | 14d |
| Evaluación | 3d |
| Preparación Demo | 3d |
| Documentación del proyecto | 142d |
| Preparación de la presentación del proyecto | 2d |
| Reunión 1 | 1h |
| Reunión 2 | 1h |
| Reunión 3 | 1h 30... |
| Reunión 4 | 1h |
| Reunión 5 | 1h |
| Reunión 6 | 2h |
| Reunión 7 | 1h 30... |
| Reunión 8 | 2h 30... |
| Reunión 9 | 2h 30... |
| Reunión 10 | 2h 30... |
| Reunión 11 | 2h |
| Reunión 12 | 1h 30... |





| Nombre | Trab... |
|---|---|
| Instalación compilador desde código fuente | 4d |
| Inicio configuración kernel | 24d |
| Instalación compilador del sistema | 10d |
| Instalación programas básicos | 5d |
| **Implementación de la versión 1** | **31d** |
| Instalación programas complementarios | 24d |
| Arranque desde USB (sin run-levels) | 7d |
| Configurar y compilar kernel | 24d |
| **Implementación de la versión 2** | **44d** |
| Instalación paquetes arranque+montaje | 4d |
| Arranque desde USB (con run-levels) | 2d |
| Instalación programas complementarios | 2d |
| Conectar a Internet | 1d |
| Sistema autosuficiente | 6d |
| Entorno gráfico | 15d |
| Monitorización de servicios | 14d |
| Evaluación | 3d |
| Preparación Demo | 3d |
| Documentación del proyecto | 142d |
| Preparación de la presentación del proyecto | 2d |
| Reunión 1 | 1h |
| Reunión 2 | 1h |
| Reunión 3 | 1h 30... |
| Reunión 4 | 1h |
| Reunión 5 | 1h |
| Reunión 6 | 2h |
| Reunión 7 | 1h 30... |
| Reunión 8 | 2h 30... |
| Reunión 9 | 2h 30... |
| Reunión 10 | 2h 30... |
| Reunión 11 | 2h |
| Reunión 12 | 1h 30... |



| Nombre | Trab... |
|---|---|
| Instalación compilador desde código fuente | 4d |
| Inicio configuración kernel | 24d |
| Instalación compilador del sistema | 10d |
| Instalación programas básicos | 5d |
| **Implementación de la versión 1** | **31d** |
| Instalación programas complementarios | 24d |
| Arranque desde USB (sin run-levels) | 7d |
| Configurar y compilar kernel | 24d |
| **Implementación de la versión 2** | **44d** |
| Instalación paquetes arranque+montaje | 4d |
| Arranque desde USB (con run-levels) | 2d |
| Instalación programas complementarios | 2d |
| Conectar a Internet | 1d |
| Sistema autosuficiente | 6d |
| Entorno gráfico | 15d |
| Monitorización de servicios | 14d |
| Evaluación | 3d |
| Preparación Demo | 3d |
| Documentación del proyecto | 142d |
| Preparación de la presentación del proyecto | 2d |
| Reunión 1 | 1h |
| Reunión 2 | 1h |
| Reunión 3 | 1h 30... |
| Reunión 4 | 1h |
| Reunión 5 | 1h |
| Reunión 6 | 2h |
| Reunión 7 | 1h 30... |
| Reunión 8 | 2h 30... |
| Reunión 9 | 2h 30... |
| Reunión 10 | 2h 30... |
| Reunión 11 | 2h |
| Reunión 12 | 1h 30... |

| Nombre | Trab... |
|---|---|
| Instalación compilador desde código fuente | 4d |
| Inicio configuración kernel | 24d |
| Instalación compilador del sistema | 10d |
| Instalación programas básicos | 5d |
| **Implementación de la versión 1** | **31d** |
| Instalación programas complementarios | 24d |
| Arranque desde USB (sin run-levels) | 7d |
| Configurar y compilar kernel | 24d |
| **Implementación de la versión 2** | **44d** |
| Instalación paquetes arranque+montaje | 4d |
| Arranque desde USB (con run-levels) | 2d |
| Instalación programas complementarios | 2d |
| Conectar a Internet | 1d |
| Sistema autosuficiente | 6d |
| Entorno gráfico | 15d |
| Monitorización de servicios | 14d |
| Evaluación | 3d |
| Preparación Demo | 3d |
| Documentación del proyecto | 142d |
| Preparación de la presentación del proyecto | 2d |
| Reunión 1 | 1h |
| Reunión 2 | 1h |
| Reunión 3 | 1h 30... |
| Reunión 4 | 1h |
| Reunión 5 | 1h |
| Reunión 6 | 2h |
| Reunión 7 | 1h 30... |
| Reunión 8 | 2h 30... |
| Reunión 9 | 2h 30... |
| Reunión 10 | 2h 30... |
| Reunión 11 | 2h |
| Reunión 12 | 1h 30... |

# 11. Costes del proyecto

Los costes de este proyecto pueden dividirse en tres apartados:

Costes del hardware

Ordenador portátil: 800 €
PC de sobremesa: 400 €
Pendrive 4 GB: 10 €
Router: 20 €

Costes del software

Todo el software utilizado en este proyecto es libre: Linux, los paquetes usados de GNU, y los non-GNU, tienen licencia GPL, que indica que pueden usarse libremente sin ningún coste.

Costes del personal

| Perfil | Precio Hora | Jornadas | Dedicación | Coste |
|---|---|---|---|---|
| Ingeniero | 40 € | 105 | 100 % | 18900 € |

nota: una jornada equivale a una media de 4.5 horas trabajadas por día laborable

Teniendo en cuenta el coste del material, software y el coste de personal, el coste total de este proyecto es de 20,130 €.

# 12. Conclusiones

En este apartado se presentan las conclusiones que se han obtenido durante el desarrollo de este proyecto.

## 12.1. Objetivos conseguidos

Durante el proyecto, se han conseguido la mayoría de los objetivos propuestos, excluyendo la configuración del *kernel* de Linux, que se ha quedado aproximadamente al 80%, y el uso de los contenedores (comentado en el informe previo). Este último tema ha sido substituido por la instalación de un entorno de monitorización de sistemas y redes.

Además de los objetivos marcados, en la versión 2 también he compilado e instalado un servidor X para poder utilizar un entorno de escritorio, y de esta manera poder tener también un programa de monitorización. Este programa se puede controlar a través de un navegador web y se utiliza un servidor web para dar formato a la información obtenida durante la monitorización.

## 12.2. Valoración personal

Siempre he querido saber de una manera más práctica (comparando con la manera en que se nos explica en la carrera), cómo es la estructura de un sistema operativo. Por ejemplo, qué hace para arrancar, cómo se pueden modificar los pasos que sigue al iniciar el sistema, etc; para saber cómo administrarlo correctamente.

Es por estas razones por las que finalmente elegí construir un sistema Linux a partir de

código fuente, instalando y configurando los paquetes necesarios para la monitorización de servicios y para poder arreglar ciertos aspectos de otras máquinas mediante éste.

Una vez acabado, puedo decir que se ha conseguido lo que se esperaba del proyecto, he obtenido un aprendizaje de SO a un nivel más bajo del habitual. También he aprendido mucho sobre configuración, compilación e instalación de paquetes, lo que me ayudado también a conocer más el sistema operativo, por la necesidad de tener que buscar ciertos archivos como cabeceras o bibliotecas para determinadas instalaciones, y consecuentemente me ha ayudado también, con el aprendizaje de la administración de un sistema; también he obtenido bastantes conocimientos sobre programas para arreglar otros equipos y de monitorización, junto con su funcionamiento; y por último, comprensión de aspectos del *kernel* de Linux y como configurarlo, compilarlo e instalarlo, ya que, aunque no haya podido completar la configuración al 100%, me he quedado cerca del final, y he comprobado cómo se compila y se instala, dejando algunos valores por defecto, y que funciona como lo hacía el que ya tenía anteriormente.

# 13. Referencias

[1] http://gcc.gnu.org/install/

[2] http://www.linux-cd.com.ar/manuales/lfs-es-html/chapter06/glibc.html

[3] http://www.gnu.org

[4] http://www.

[5] http://tldp.org/FAQ/Linux-FAQ/x-windows.html

[6] http://www.x.org/wiki

[7] http://www.xfree86.org/4.8.0/BUILD.html

[8] http://fluxbox-wiki.org/index.php/Build_fluxbox_from_source

[9] http://nagios.sourceforge.net/docs/3_0/quickstart-ubuntu.html

[10] http://www.

net

# Anexo A - Configuración, compilación e instalación del kernel

Los primeros pasos los haremos en el sistema Ubuntu local, no en la máquina virtual, ya que al compilar, ocupa bastante espacio y podría ser que no tuviéramos el suficiente en la máquina virtual. Y una vez tengamos el kernel compilado, Lo pasaremos a la máquina virtual donde lo instalaremos para ver que funciona correctamente y así poder ponerlo también en el pendrive.[13]

Me descargo el último kernel estable de www.kernel.org, que en mi caso es la versión 2.6.39.1.

descomprimir:
tar xvfj linux-2.6.39.1.tar.bz2;

Entramos en la carpeta linux-2.6.39.1. Si se hubiera compilado alguna vez, habría que borrar todos los archivos de configuración haciendo:
make mrproper;
(Como nos lo acabamos de descargar, no hay que hacerlo)

Configuración:
make menuconfig;

Vemos que nos hacen falta las bibliotecas ncurses, así que las instalamos instalamos:
sudo apt-get install libncurses5-dev;

Como ya he explicado anteriormente en este proyecto, la configuración no está completa, no queda mucho, está alrededor del 80%, quedan algunos apartados por mirar. Con esta configuración parcial y el resto por defecto el kernel funciona correctamente, ya que es el

ha sido usado en la versión 2.

Para mostrar como configurar el kernel, se van a ir viendo los diferentes apartados seguidos de una breve explicación, y la decisión de si el apartado es incluido o no.

Configuración (Y = incluir; N = no incluir; M = incluir en módulo(se usa cuando haga falta)):
General setup:

- Prompt for development and/or incomplete code/drivers: para usar drivers que no estén finalizados del todo (que sean beta): Y
- Cross-compiler tool prefix: compilación cruzada: “ “
- Local version - append to kernel release: personalizar este kernel con mi “firma” y así darle un nombre único: “ “
- Automatically append version information to the version string: sirve para lo mismo que el anterior: N
- Kernel compression mode: elegir la forma en que se comprimirá: Gzip
- Support for paging of anonymous memory: permite usar swap: Y
- System V IPC: permite la intercomunicación de los procesos, por lo tanto siempre tiene que estar activa: Y
- POSIX Message Queues: Y
- BSD Process Accounting: permite a los administradores saber que hacen los usuarios en todo momento en el sistema: N
    - BSD Process Accounting version 3 file format: guarda la información en un nuevo formato de archivo que guarda la id del proceso y la del padre (se necesita la versión 3 de BSD): N
- open by fhandle syscalls: N
- Export task/process statistics through netlink: envía estadísticas a través de la interfaz de red genérica, al contrario que BSD, solo lo puede hacer mientras las tareas o procesos no han finalizado: N
- Enable system-call auditing support: N
- IRQ subsystem:
    - Support sparse irq numbering: N
- RCU subsystem:
    - RCU Implementation: Tree-based hierarchical RCU
    - Enable tracing for RCU: Da información al debuguear implementaciones en

RCU: N
    - Tree-based hierarchical RCU fanout value: 32
    - Disable tree-based hierarchical RCU auto-balancing: N
    - Accelerate last non-dyntick-idle CPU's grace periods: acelera “el uso” de CPU para ponerse antes en estado “iddle”(eficiencia energética): N
- Kernel .config support: copiar el fichero de configuración .config dentro del mismo kernel (copia de seguridad) por si se perdiera el archivo, poder recuperarlo: M
- Kernel log buffer size: selecciona el tamaño del buffer de log del kernel en potencias de 2 (12 => 4KB, 13 => 8KB, … , 16 => 64KB, 17 => 128KB, ...): 12
- Control Group support:
    - Example debug cgroup subsystem: activa un subsistema cgroup que exporta información de debugueo sobre cgroups framework: N
    - Namespace cgroup subsystem: N
    - Freezer cgroup subsystem: permite pausar y continuar todas las tareas en un cgroup: N
    - Device controller for cgroups: permite a un cgroup crear una lista de dispositivos que un proceso del cgroup puede abrir: N
    - Cpuset support: permite crear conjuntos de CPU’s que permite particionar el sistema de forma dinámica en los conjuntos de CPU’s y nodos de memoria, y asignar tareas únicamente a esos conjuntos (para sistemas SMP y NUMA): N
        - Include legacy /proc/<pid>/cpuset file: la manera de almacenar los procesos teniendo en cuenta su conjunto de CPU’s: N
    - Simple CPU accounting cgroup subsystem: provee de un controlador para monitorizar el total de CPU que utilizan las tareas en un cgroup: N
    - Resource counters: N
        - Memory Resource Controller for Control Groups: N
            - Memory Resource Controller Swap Extension: N
                - Memory Resource Controller Swap Extension enabled by default: N
    - Enable perf_event per-cpu per-container group (cgroup) monitoring: restringe la monitorización de threads de un grupo específico y que corra en la CPU designada: N
    - Group CPU scheduler:

        - Group scheduling for SCHED_RR/FIFO: permita asignear ancho de banda de CPU real a tareas de grupos (puede hacer que tareas de usuarios no se hagan en tiempo real): N
    - Block IO controller: interfaz de control de E/S de los cgroups: N
        - Enable Block IO controller debugging: crea ficheros de estado en los cgroups que son útiles para debuguear: N
- Namespaces support: N
    - UTS namespace: las tareas ven diferente información de la llamada de sistema uname(): N
    - IPC namespace: N
    - User namespace: permite contenedores: N
    - PID Namespaces: Support process id namespaces, permite tener varios procesos con el mismo pid si tienes diferente pid de namespace. Contrucción de bloques de contenedores: N
    - Network namespace: N
- Automatic process group scheduling: optimiza la “programación” de tareas de un escritorio común creando grupos de tareas automáticamente: Y
- Enable deprecated sysfs features to support old userspace tools: se usaba en versiones anteriores por el cambio que se hizo de almacenar en /sys/class/block/ a /sys/block/: N
- Initial RAM filesystem and RAM disk (initramfs/initrd) support: en principio es necesario para pantallazos y para drivers de RAID: Y
    - Initramfs source file(s): ubicación de los archivos que forman initramfs: “ “
    - Support initial ramdisks compressed using gzip: soporta la codificación gzip para ramdisk inicial: Y
    - Support initial ramdisks compressed using bzip2: Y
    - Support initial ramdisks compressed using LZMA: Y
    - Support initial ramdisks compressed using XZ: Y
    - Support initial ramdisks compressed using LZO: Y
- Optimize for size: reduce el tamaño final del kernel, pero se han dado casos de inestabilidad del SO y el kernel ocupa poco teniendo en cuenta los discos duros de hoy en día: N
- Configure standard kernel features: configurar opciones básicas del kernel: Y
    - Enable 16-bit UID system calls: útil para casos especiales: Y

- Sysctl syscall support: útil para casos especiales: Y
- Load all symbols for debugging/ksymoops:
  - Include all symbols in kallsyms: mostrar mensajes en fallos del kernel: Y
  - Do an extra kallsyms pass: si no se muestran mensajes cuando hay fallo del kernel hay que activar esta: N
- Support for hot-pluggable devices: no se necesita si no se necesitan módulos, como en un embedded system: Y
- Enable support for printk: muestra mensajes del kernel, útil para diagnosticar problemas del kernel: Y
- BUG() support: da soporte para bugs, solo se elimina si se utiliza un sistema que no tiene facilidad en reportar errores (algunos embedded systems): Y
- Enable ELF core dumps: activa el core dump: Y
- Enable PC-Speaker support: activa pc-speaker: Y
- Enable full-sized data structures for core: Y
- Enable futex support: activa “fast userspace mutexes”, necesario para cargar correctamente las aplicaciones “glibc-based”: Y
- Enable eventpoll support: activa el mecanismo de notificación de E/S epoll que es útil porque mejora el rendimiento de las llamadas de sistema cuando hay muchos file descriptors de O(n) a O(1): Y
- Enable signalfd() system call: activa la llamada de sistema signalfd que permite recibir signals a un file descriptor: Y
- Enable timerfd() system call: activa a llamada de sistema timerfd que permite recibir eventos del timer a un file descriptor: Y
- Enable eventfd() system call: activa a llamada de sistema eventfd que permite recibir notificaciones del kernel o del userspace: Y
- Use full shmem filesystem: es un sistema de ficheros interno que organiza la memoria compartida: Y
- Enable AIO support: activa las E/S asincronas de POSIX, es usado por aplicaciones de alto rendimiento con threads: Y

- Embedded system: N
- Kernel Performance Events And Counters:
  - Kernel performance events and counters: permite al kernel soportar varios eventos de rendimiento del software y el hardware: Y

    - Kernel performance counters (old config option): está obsoleto por PERF_EVENTS, solo se usa por compatibilidad: N
    - Debug: use vmalloc to back perf mmap() buffers: usar vmalloc (library for dynamic memory allocation) para volver a nmap() perf buffers: N
- Enable VM event counters for /proc/vmstat: activa un contador de enventos para vmstat: Y
- Enable PCI quirk workarounds: N
- Enable SLUB debugging support: añade características de debugueo, que no nos sirven porque en caso de fallar el sistema no podremos encontrar el error debugueando y al desactivarlo reducimos el tamaño del código: N
- Disable heap randomization: para activar la “heap randomization”: N
- Choose SLAB allocator: escogemos la más rápida: SLUB (Unqueued Allocator)
- Profiling support: activa mecanismos extendidos de soporte para perfiles que son usados por herramientas como Oprofile: Y
- OProfile system profiling: activar Oprofile: M
    - OProfile multiplexing support: permite generar más eventos que contadores proporcionados por el hardware: N
- Kprobes: Kprobes permite capturar direcciones del kernel y ejecutar una función de retorno, es útil para debuguear y testear: N
- Optimize trace point call sites: si el compilador soporta “asm goto” -> el kernel compila los trace points con una instrucción nop, pasa a poner un jump a la función trace. Reduce la carga en la rama de precisión del procesador: Y
- GCOV-based kernel profiling:
    - Enable gcov-based kernel profiling: N

Enable loadable module support: elegir módulos que se cargarán: Y

- Forced module loading: fuerza la carga de módulos: N
- Module unloading: permite dejar de usar módulos que están siendo usados en memoria: Y
    - Forced module unloading: fuerza dejar de usar los módulos: N
- Module versioning support: permite usar módulos compilados por un kernel de una versión diferente a esta: N (solo dejamos usar módulos hechos para esta versión)
- Source checksum for all modules: es usada para problemas de arraque de los módulos, hace un checksum de los ficheros de código que se usan para montar el

módulo (en el caso en que sea de otra versión de kernel no dejará montarlo porque fallará el checksum): Y

Enable the block layer: Y

- Support for large (2TB+) block devices and files: sirve para poder soportar bloques muy grandes de los dispositivos (discos, RAID, loopback...). Es necesario para el sistema de archivos ext4: Y
- Block layer SG support v4: activa el SCSI genérico v4. Lo necesitan las versiones recientes del gestor de dispositivos del kernel de linux (udev): Y
- Block layer data integrity support: proporciona herramientas a varios sistemas de archivos para asegurar la integridad de los datos de algunos dispositivos de almacenamiento: Y
- Block layer bio throttling support: permite limitar el ratio de E/S de los cgroups a un dispositivo: Y
- IO Schedulers:
    - Deadline I/O scheduler: proporciona un servicio CSCAN con una planificación FIFO de las peticiones: M
    - CFQ I/O scheduler: distribuye el ancho de banda entre todos los procesos del sistema: Y
    - Default I/O scheduler (CFQ): CFQ (porque es el que se usa por defecto y es reciente (2008))

Processor type and features:

- Tickless System (Dynamic Ticks): la interrupción de reloj solo se activa cuando es necesaria: Y
- High Resolution Timer Support: alta resolución del timer: Y
- Symmetric multi-processing support: soporte para que el sistema pueda trabajar con multiprocesadores en vez de en monoprocesador: Y
- Enable MPS table: para sistemas antiguos que no tienen soporte acpi: N
- Support for big SMP systems with more than 8 CPUs: soporte para sistemas con más de 8 CPUs: N
- Support for extended (non-PC) x86 platforms: esta opción es por si se tiene un sistema poco común ( AMD Elan, NUMAQ (IBM/Sequent), RDC-321x SoC, SGI 320/540 (Visual Workstation), Summit/EXA (IBM x440), Unisys ES7000 IA32 series,

Moorestown MID devices) ): N

- Eurobraille/Iris poweroff module: proporcionan comandos para apagar a las maquinas IRIS: M
- Single-depth WCHAN output: calcula el valor de WCHAN en /proc/<PID>/wchan: Y
- Paravirtualized guest support: soporte para virtualización (máquinas virtuales): N
- paravirt-ops debugging: debug para máquinas virtuales: N
- Memtest: añade el parámetro memtest al kernel: Y
- Processor family: Pentium-Pro (el más reciente)
- Generic x86 support: incluye optimizaciones para procesadores x86: Y
- PentiumPro memory ordering errata workaround: en los antiguos multiprocesadores PentiumPro había un error que muy pocas veces hacía una violación de acceso a memoria: N
- Supported processor vendors: que código de los diferentes vendedores de procesadores acepta: Y
    - Support Intel processors: Y
    - Support Cyrix processors: Y
    - Support AMD processors: Y
    - Support Centaur processors: Y
    - Support Transmeta processors: Y
    - Support UMC processors: Y
- HPET Timer Support: HPET es la nueva genración del timer que reemplaza la de 8254s: Y
- Enable DMI scanning: activa la identificación de la máquina por DMI: Y
- Maximum number of CPUs: 8
- SMT (Hyperthreading) scheduler support: Y
- Multi-core scheduler support: Y
- Fine granularity task level IRQ time accounting: activa el timstamp entre transiciones de estado de irq: N
- Preemption Model: Voluntary Kernel Preemption (la opción para equipos normales)
- Reroute for broken boot IRQs: Y
- Machine Check / overheating reporting: permite al procesador avisar al kernel cuando detecta errores: Y
    - Intel MCE features: Y
    - AMD MCE features: Y

    - Support for old Pentium 5 / WinChip machine checks: N
- Machine check injector support: revisa la máquina para testeos: M
- Enable VM86 support: es necesario para programas que utilizan la una política de 16-bits: Y
- Toshiba Laptop support: M
- Dell laptop support: M
- Enable X86 board specific fixups for reboot: permita utilizar el comando reboot correctamente para las máquinas utilicen los chipsets CS5530A y CS5536 y el RDC R-321x SoC: N
- /dev/cpu/microcode - microcode support: permite actualizar el microcódigo de algunos procesadores de Intel y AMD: M
    - Intel microcode patch loading support: Y
    - AMD microcode patch loading support: Y
- /dev/cpu/*/msr - Model-specific register support: permite a los procesos privilegiados acceder a los MSRs: M
- /dev/cpu/*/cpuid - CPU information support: permite a un proceso ejecutar una determinada instrucción en un procesador determinado: M
- High Memory Support: 64GB
- Memory split: 3G/1G user/kernel split
- PAE (Physical Address Extension) Support: es necesario para soporte NX: Y
- Memory model: Flat Memory
- Allow for memory compaction: permite compactar la memoria para alojar páginas de memoria grandes: Y
- Page migration: Y
- Enable KSM for page merging: si encuentra 2 páginas con el mismo contenido, las unifica para tener más memoria libre: Y
- Low address space to protect from user allocation: 65536
- Enable recovery from hardware memory errors: Y
    - HWPoison pages injector: M
- Transparent Hugepage Support: permite al kernel que use grandes páginas de memoria y un TLB grande de manera transparente a las aplicaciones (evita fallos de página): N
- Allocate 3rd-level pagetables from highmem: pone la tabla de páginas de usuario en memoria si es suficientemente grande: Y

- Check for low memory corruption: Y
    - Set the default setting of memory_corruption_check: Y
- Amount of low memory, in kilobytes, to reserve for the BIOS: 64
- Math emulation: simula un procesador matemático si no dispone de uno: N
- MTRR (Memory Type Range Register) support: registros usados en la familia P6 de Intel: Y
    - MTRR cleanup support: Y
        - MTRR cleanup enable value (0-1): 0
        - MTRR cleanup spare reg num (0-7): 1
    - x86 PAT support: Y
- EFI runtime service support: útil si el sistema usa firmware EFI: Y
- Enable seccomp to safely compute untrusted bytecode: es útil para aplicaciones que utilizan bytecode que no es de confianza, permitiédole utilizar únicamente algunas llamadas a sistema que sean seguras: Y
- Enable -fstack-protector buffer overflow detection: activa el flag -fstack-protector de GCC. Coloca un valor, al llamar a las funciones, antes de la dirección de retorno, y lo valida antes del retorno. “Stack based buffer overflow” sobreescribe ahora este valor, y el kernel puede detectar el ataque y neutralizarlo: Y
- Timer frequency: 250 HZ
- kexec system call: Es una especie de reboot independiente del firmware y que sirve solo para cambiar el kernel utilizado: N
- kernel crash dumps: genera un volcado después de iniciar el sistema con la llamada kexec: N
- kexec jump: hace la transición entre el viejo kernel y el nuevo kernel cargado y coloca código en direcciones físicas a través de kexec: N
- Physical address where the kernel is loaded: 0x1000000
- Build a relocatable kernel: crea la imagen del kernel con información de reubicación. Hace el tamaño de la imagen un 10% más grande. Es útil, entre otras cosas, para la llamada kexec, que tiene que cargar el kernel temporalmente en una dirección de memoria diferente a la del kernel principal: N
- Alignment value to which kernel should be aligned: 0x1000000
- Support for hot-pluggable CPUs: permite cambiar la CPU también cuando está encendida la máquina: Y
- Compat VDSO support: Compatibilidad con un sistema de direcciones usado

antiguamente. Necesario si nuestra glibc es anterior a la 2.3.3: N

- Built-in kernel command line: permite cambiar algunos parámetros de la configuración al compilar el kernel: N

Power management and ACPI options:

- Suspend to RAM and standby: permite entrar en modo suspensión: Y
- Hibernation: Permite utilizar el modo hibernación: Y
    - Default resume partition: es la partición donde el modo hibernación mirará al salir de este modo: “ “
- Run-time PM core functionality: hace que los dispositivos de entrada salida entren en un modo de bajo coste de energía después de un tiempo de inactividad, y se recuperen cuando se le haga alguna petición a su driver: Y
- Power Management Debug Support: Activa soporte para debuguear en temas relacionados con la administración de energía: N
- Suspend/resume event tracing: más opciones de debuguear relacionadas con el modo suspensión: N
- ACPI (Advanced Configuration and Power Interface) Support: Y
    - Deprecated /proc/acpi files: es por compatibilidad. Permite que exista el archivo /proc/acpi aunque su función haya sido sustituida por /sys: Y
    - Deprecated power /proc/acpi directories: para lo mismo que la opción anterior pero trantando directorios de acpi: Y
    - ACPI 4.0 power meter: necesario si tenemos un firmware ACPI 4.0: M
    - EC read/write access through /sys/kernel/debug/ec: activa opciones de debugueo del kernel: N
    - Deprecated /proc/acpi/event support: algo parecido a las 2 primeras opciones. Compatibilidad con un sistema que utilizaba ACPI de generador de eventos: Y
    - AC Adapter: adaptador de corriente. Si el sistema puede funcionar tanto con batería como con adaptador de corriente: M
    - Battery: aporta información sobre la batería en sistemas que la utilizan: M
    - Button: añade soporte de botones que sirven para apagar, entrar en modo suspensión, subir/bajar volumen... : M
    - Video: añade un driver para los gráficos integrados en la placa madre: M
    - Fan: añade drivers para controlar la refrigeración: M

    - Dock: Y
    - Processor: sirve para ahorrar energía cuando el procesador no está haciendo nada (está en estado “iddle”): Y
    - IPMI: este driver permite acceder al controlador del BMC(*baseboard management controller*) que administra la comunición entre el software y el hardware: M
    - Processor Aggregator: permite controlar todos los procesadores en ACPI 4.0: M
    - Thermal Zone: evita que los procesadores se dañen por demasiada calor: Y
    - Custom DSDT Table file to include: “ “
    - Disable ACPI for systems before Jan 1st this year: Desactiva que ACPI esté instaurado por defecto a partir del año indicado: 2000
    - Debug Statements: los subsistemas de ACPI puede debuguear: N
    - PCI slot detection driver: crea archivos para indicar que ha detectado buses PCI: M
    - Power Management Timer Support: añade un timer a los buses PCI que utilizan los sistemas actuales: Y
    - Container and Module Devices: añade soporte para los contenedores y los módulos de dispositivos de ACPI: Y
    - Smart Battery System: driver que permite acceder a la información de la batería de una manera diferente que se encuentra en algunos portátiles: M
    - Hardware Error Device: informa sobre errores de harware vía SCI (“Serial Communications Interface”): M
    - ACPI Platform Error Interface (APEI): permite avisar al sistema operativo de los errores: Y
        - APEI Generic Hardware Error Source: M
        - APEI PCIe AER logging/recovering support: N
        - APEI Error INJection (EINJ): se usa para debugueo: N
        - APEI Error Record Serialization Table (ERST) Debug Support: N
- SFI (Simple Firmware Interface) Support: provee de un méodo para enviar información al sistema operativo mediante una tablas estáticas en memoria: Y
- APM (Advanced Power Management) BIOS support: diferentes técnicas de ahorro de energía: M
- CPU Frequency scaling: Y

- Enable CPUfreq debugging: N
- CPU frequency translation statistics: genera información estadística sobre la frecuencia de la CPU: M
  - CPU frequency translation statistics details: Y
- Default CPUFreq governor: ondemand
- 'performance' governor: esta configuración de CPU pone la frecuencia a lo más alto posible: Y
- 'powersave' governor: lo mismo pero al mínimo posible: M
- 'userspace' governor for userspace frequency scaling: ésta permite poner la frecuencia deseada manualmente: M
- 'ondemand' cpufreq policy governor: en este caso pone la frecuencia que se necesita en cada momento: Y
- 'conservative' cpufreq governor: similar a ‘ondemand’ pero optimizada para que la batería de los portatiles dure más: M
- Processor Clocking Control interface driver: M
- ACPI Processor P-States driver: M
- AMD Mobile K6-2/K6-3 PowerNow!: M
- AMD Mobile Athlon/Duron PowerNow!: M
- AMD Opteron/Athlon64 PowerNow!: M
- Cyrix MediaGX/NatSemi Geode Suspend Modulation: M
- Intel Speedstep on ICH-M chipsets (ioport interface): M
- Intel SpeedStep on 440BX/ZX/MX chipsets (SMI interface): M
- Intel Pentium 4 clock modulation: M
- nVidia nForce2 FSB changing: M
- Transmeta LongRun: M
- VIA Cyrix III Longhaul: M
- VIA C7 Enhanced PowerSaver: M
- Relaxed speedstep capability checks: no realiza todos los controles que un sistema con cambio de velocidad haría: Y

- CPU idle PM support: soporte para que el estado iddle del procesador consuma menos mediante software: Y
  - Cpuidle Driver for Intel Processors: incluye conocimiento sobre caraterísticas iddle de hardware de intel: Y

Bus options (PCI etc.):

- PCI support: soporte para una placa base que utiliza buses PCI: Y
    - PCI access mode: Any
    - Read CNB20LE Host Bridge Windows: permite PCI hotplug en sistemas que tiene un chipset CNB20LE sin ACPI: N
- Support for DMA Remapping Devices: activa el direccionamiento independiente a DMA por parte de los dispositivos: Y
    - Enable DMA Remapping Devices by default: N
- PCI Express support: Y
    - PCI Express Hotplug driver: M
    - Root Port Advanced Error Reporting support: N
    - PCI Express ASPM control: activa control sobre ahorro de energía: Y
        - Debug PCI Express ASPM: N
- Message Signaled Interrupts (MSI and MSI-X): permite a los drivers de los dispositivos activar MSI(Message Signaled Interrupts): Y
- PCI Debugging: N
- PCI Stub driver: reserva un dispositivo PCI cuando va a ser asignado a un huésped del sistema operativo: M
- Xen PCI Frontend: M
- Interrupts on hypertransport devices: permite las interrupciones de los dispositivos hypertransport: Y
- ISA support: da soporte por si nuestra placa base utiliza buses ISA: Y
    - EISA support: da soporte por si nuestra placa base utiliza buses EISA: Y
        - Vesa Local Bus priming: Y
        - Generic PCI/EISA bridge: Y
        - EISA virtual root device: Y
        - EISA device name database: Y
- MCA support: soporte por si nuestra placa base utiliza buses MCA: Y
    - Legacy MCA API Support: soporte para antiguas placas MCA API: Y
        - Support for the mca entry in /proc: utilizar el viejo como /proc/mca en vez del de los nuevos sistemas de ficheros: N
- NatSemi SCx200 support: proporciona soporte para algunos tipos de procesadores: M
    - NatSemi SCx200 27MHz High-Resolution Timer Support: soluciona el

problema de que algún procesador de ese tipo perdía la hora cuando el procesador entraba en estado iddle: M

- PCCard (PCMCIA/CardBus) support: M
    - Todos sus subapartados hay que ponerlos como M si se puede, y si no como Y.
- Support for PCI Hotplug: soporte hotplug para tarjetas PCI si la placa base lo permite: M
    - Todos sus subapartados hay que ponerlos como M si se puede, y si no como Y.
- RapidIO support: RapidIO son interconexiones en embedded systems que va muy rápido: N

Executable file formats / Emulations:

- Kernel support for ELF binaries: ELF (Executable and Linkable Format) es un formato de bibliotecas o ejecutables que sirven para diferentes arquitecturas y sistemas operativos: Y
- Write ELF core dumps with partial segments: escribe un mapeo de memoria de procesos caídos: N
- Kernel support for a.out and ECOFF binaries: son formatos de bibliotecas o ejecutables que se utilizaban en la primeras version de UNIX: M
- Kernel support for MISC binaries: M

Networking support:

- Networking options:
    - Packet socket: Y
    - Unix domain sockets: Y
    - Transformation user configuration interface: M
    - Transformation sub policy support: poder usar más de una política a un paquete: N
    - Transformation migrate database: necesario para IPv6: N
    - Transformation statistics: estadísticas sobre errores en la transformación de paquetes: N
    - PF_KEY sockets: M
        - PF_KEY MIGRATE: N

- TCP/IP networking: Y
  - IP: multicasting: Y
  - IP: advanced router: Utilizar el sistema como un router: Y
    - FIB TRIE statistics: útil para testeo de TRIE: N
    - IP: policy routing: permite al sistema acceder a la dirección de origen del paquete para decidir mejor el destino: Y
    - IP: equal cost multipath: permite especificar diferentes caminos para los paquetes: Y
    - IP: verbose route monitoring: Y
  - IP: kernel level autoconfiguration: activa la configuración automática de direcciones IP al arrancar el sistema. Se necesita para máquinas que van sin disco y requieren conectarse al arrancar: N
  - IP: tunneling: M
  - IP: GRE demultiplexer: M
    - IP: GRE tunnels over IP: M
      - IP: broadcast GRE over IP: Y
  - IP: multicast routing: para máquinas que actúan como routers con paquetes con muchas destinaciones: N
  - IP: ARP daemon support: sustituto del kernel para la tabla ARP: N
  - IP: TCP syncookie support: previene ataques “flooding”: Y
  - IP: AH transformation: M
  - IP: ESP transformation: M
  - IP: IPComp transformation: M
  - IP: IPsec transport mode: M
  - IP: IPsec tunnel mode: M
  - IP: IPsec BEET mode: M
  - Large Receive Offload (ipv4/tcp): M
  - INET: socket monitoring interface: M
  - TCP: advanced congestion control: Y
    - Todos sus subapartados como M.
  - TCP: MD5 Signature Option support (RFC2385): proporciona protección MD5 a sesiones TCP: Y
  - The IPv6 protocol: M
    - Todos los subapartados los dejamos por defecto (todos los

posibles a módulos).

- NetLabel subsystem support: soporte para protocolos CIPSO y RIPSO: Y

- Security Marking: Y
- Timestamping in PHY devices: añade información el los envíos de paquetes por PHY: N
- Network packet filtering framework (Netfilter): Net filter sirve para controlar el envío de paquetes por la red: Y
  - Network packet filtering debugging: N
  - Advanced netfilter configuration: N
  - Core Netfilter Configuration:
    - Netfilter LOG over NFNETLINK interface: M
    - Netfilter connection tracking support: guarda los paquetes que han pasado a través de la máquina: M
      - Connection tracking security mark support: Para seguridad en las conexiones: Y
      - FTP protocol support: M
      - IRC protocol support: M
      - SIP protocol support: M
      - Connection tracking netlink interface: M
    - Netfilter Xtables support (required for ip_tables): M
    - El resto todo en M que es soporte para algunas funcionalidades
  - IP set support: N
  - IP virtual server support: N
  - IP: Netfilter Configuration: son características, lo ponemos todo a M sis se puede y si no a Y
  - IPv6: Netfilter Configuration: similar al anterior
- The DCCP Protocol: Datagram Congestion Control Protocol, es útil para cosas como juegos online o ver vídeos en internet: N
- The SCTP Protocol: Stream Control Transmission Protocol, ponemos todos sussubapartados, que son para debugueo, a N
- The RDS Protocol: Reliable Datagram Sockets, proporciona fiabilidad para las secuencias de datagramas: N

- The TIPC Protocol: Transparent Inter Process Communication, para comunicaciones dentro de clústeres: N
- Asynchronous Transfer Mode (ATM): tecnología de red de alta velocidad: M
  - Classical IP over ATM: M
    - Do NOT send ICMP if no neighbour: N
  - LAN Emulation (LANE) support: emula servicios para LAN's en redes ATM: M
    - Multi-Protocol Over ATM (MPOA) support: M
  - RFC1483/2684 Bridged protocols: las comunicaciones ATM actúan como una Ethertnet a vista del kernel: M
    - Per-VC IP filter kludge: N
- Layer Two Tunneling Protocol (L2TP): da facilidades al tráfico de red: M
  - L2TP debugfs support: N
  - L2TPv3 support: N
- 802.1d Ethernet Bridging: permite a la máquina actuar cmo un bridge: M
  - IGMP/MLD snooping: Y
- Distributed Switch Architecture support: permite utilizar chips hardware que tengan arquitectura de switch: Y
  - Marvell 88E6060 ethernet switch chip support: Y
  - Marvell 88E6085/6095/6095F/6131 ethernet switch chip support: Y
  - Marvell 88E6123/6161/6165 ethernet switch chip support: Y
- 802.1Q VLAN Support: M
  - GVRP (GARP VLAN Registration Protocol): Y
- DECnet Support: M
  - DECnet: router support: M
- ANSI/IEEE 802.2 LLC type 2 Support: soporte para determinados sockets: M
- The IPX protocol: M
  - IPX: Full internal IPX network: N
- Appletalk protocol support: protocolo de red de los ordenadores Apple: N
- CCITT X.25 Packet Layer: grupo de protocolos de red estándar: M
- LAPB Data Link Driver: fiabilidad de conexión para X.25: M
- Acorn Econet/AUN protocols: protocolo de red para ordenadores Acorn: N
- WAN router: permite ser un router de redes WAN (varias redes LAN): M
- Phonet protocols family: protocolo para modems Nokia: N

    - IEEE Std 802.15.4 Low-Rate Wireless Personal Area Networks support: N
    - QoS and/or fair queueing: planificador de las colas de paquetes de red: Y
        - Todos los subapartados por defecto
    - Data Center Bridging support: Y
    - B.A.T.M.A.N. Advanced Meshing Protocol: N
    - Network testing: todo a N
- Amateur Radio support: ...

Para poder compilarlo nos hace falta un “generador de paquetes”:
sudo apt-get install kernel-package

Compilación:
make-kpkg clean
make-kpkg --initrd kernel_image kernel_headers

Instalación:
dpkg -i kernel-image-2.6.39.1.deb

dpkg -i kernel-headers-2-6-39.1.deb

Una vez hecho esto, en /boot tendremos dos archivos nuevos, uno referente a la imnagen del kernel y otro a initrd. Tenemos que copiar estos dos archivos a /media/PFC/boot y modificar las entradas del grub (/media/PFC/boot/grub/grub.cfg) y añadir una similar a la que se ha creado en /boot/grub/grub.cfg.
Por último solo queda copiar los módulos de /lib/modules/2.6.39.1 a:
/media/PFC/lib/modules/2.6.39.1

# Glosario

En este apartado se explican las definiciones de algunos conceptos que aparecen durante la configuración del kernel de Linux.

cgroup: abreviado de "control group", proporciona un mecanismo para crear y manejar grupos de tareas con parentesco padre-hijos.
namespace: minimiza las colisiones de nombres.
core dump: guarda el estado de un programa en un determinado momento, normalmente cuando termina de una forma anormal.
pc-speaker: ayuda al usuario sobre errores del sistema guardando memoria sobre los errores.
OProfile: es una herramienta de supervisión de rendimiento que se ejecuta a lo largo de todo el sistema que incluye datos de modo sistema y modo usuario.
nop: no operation, instrucción en lenguaje asemblador que no hace nada.
WCHAN: dirección del kernel donde el proceso está durmiendo.
Timestamp: secuencia de caracteres, que denotan la hora y fecha (o alguna de ellas) en la cual ocurrió un determinado evento.
MSR: Model-Specific Register, registro cuyo contenido depende del modelo especÃfico de procesador.

# Anexo B - Ficheros de configuración de las herramientas de usuario

En este anexo se van a mostrar archivos de configuración los cuales por su tamaño no es conveniente poner en la explicación del proyecto.

## Nagios

- /home/pfc/usr/etc/conf.d/hosts.cfg(En mi red solo tengo dos PC's, para añadir más solo hay que copiar la estructura y poner como parent PFC):

```
define host {
 host_name  PFC
 alias      PFC
 address    192.168.1.120
 use        generic-host
}

define host {
 host_name  Other1
 alias      Other1
 address    192.168.1.10
 parents    PFC
 use        generic-host
}
```

- home/pfc/usr/etc/conf.d/hostgroups.cfg:

```
# A simple wildcard hostgroup
define hostgroup {
```

```
        hostgroup_name  all
        alias           All Servers
        members         *
        }

define hostgroup {
        hostgroup_name  localhost
        alias           Localhost
        members         PFC
        }

# A list of your Debian GNU/Linux servers
define hostgroup {
        hostgroup_name  debian-servers
        alias           Debian GNU/Linux Servers
        members         *
        }

# A list of your web servers
define hostgroup {
        hostgroup_name  http-servers
        alias           HTTP servers
        members         *
        }

# A list of your ssh-accessible servers
define hostgroup {
        hostgroup_name  ssh-servers
        alias           SSH servers
        members         *
}
```

- home/pfc/usr/etc/conf.d/services.cfg:

```
# check that web services are running
```

```
define service {
        hostgroup_name          http-servers
        service_description     HTTP
        check_command           check_http
        use                     generic-service
 notification_interval          0 ; set > 0 if you want to be renotified
}

# check that ssh services are running
define service {
        hostgroup_name          ssh-servers
        service_description     SSH
 check_command                  check_ssh!-p 22
        use                     generic-service
 notification_interval          0 ; set > 0 if you want to be renotified
}

define service {
 hostgroup_name                 localhost
 service_description            Current Load
 check_command                  check_local_load!5.0!10.0!
        use                     generic-service       ; Name of service template to use
 notification_interval          0 ; set > 0 if you want to be renotified
}

define service {
 hostgroup_name                 localhost
 service_description            Current Users
 check_command                  check_local_users!20!50
 use                    generic-service       ; Name of service template to use
        notification_interval   0 ; set > 0 if you want to be renotified
}

define service {
```

```
 hostgroup_name          localhost
 service_description     Disk Space
 check_command                   check_local_disk!20%!10%!/
        use                     generic-service         ; Name of service template to use
        notification_interval           0 ; set > 0 if you want to be renotified
}

define service {
 hostgroup_name          all
 service_description     Ping
 check_command                   check_ping!200.0,20%!600.0,60%
        use                     generic-service         ; Name of service template to use
        notification_interval           0 ; set > 0 if you want to be renotified
}
```

- home/pfc/usr/etc/conf.d/contacts.cfg:

```
define contact {
        contact_name                    root
        alias                   Root
        service_notification_period             24x7
        host_notification_period        24x7
        service_notification_options            w,u,c,r
        host_notification_options               d,r
        service_notification_commands   notify-service-by-email
        host_notification_commands      notify-host-by-email
        email                   root@localhost
}

# We only have one contact in this simple configuration file, so there is
# no need to create more than one contact group.

define contactgroup {
        contactgroup_name               admins
        alias                   Nagios Administrators
```

```
        members                 root
}
```

- home/pfc/usr/etc/conf.d/generic-host.cfg:

```
# Generic host definition template - This is NOT a real host, just a template!

define host {
        name                            generic-host ; The name of this host template
        notifications_enabled           1       ; Host notifications are enabled
        event_handler_enabled           1       ; Host event handler is enabled
        flap_detection_enabled          1       ; Flap detection is enabled
        failure_prediction_enabled      1       ; Failure prediction is enabled
        process_perf_data               1       ; Process performance data
        retain_status_information       1       ; Retain status information across program restarts
        retain_nonstatus_information    1       ; Retain non-status information across program restarts
         check_command                  check-host-alive
         max_check_attempts             10
         notification_interval          0
         notification_period            24x7
         notification_options           d,u,r
         contact_groups                 admins
        register                        0       ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL HOST, JUST A TEMPLATE!
}
```

- home/pfc/usr/etc/conf.d/generic-service.cfg:

```
# generic service template definition
define service {
        name                            generic-service ; The 'name' of this service template
        active_checks_enabled           1       ; Active service checks are enabled
        passive_checks_enabled          1       ; Passive service checks are enabled/accepted
```

```
        parallelize_check       1       ; Active service checks should be parallelized (disabling this can lead to major performance problems)
        obsess_over_service     1       ; We should obsess over this service (if necessary)
        check_freshness         0       ; Default is to NOT check service 'freshness'
        notifications_enabled   1       ; Service notifications are enabled
        event_handler_enabled   1       ; Service event handler is enabled
        flap_detection_enabled  1       ; Flap detection is enabled
        failure_prediction_enabled      1       ; Failure prediction is enabled
        process_perf_data       1       ; Process performance data
        retain_status_information       1       ; Retain status information across program restarts
        retain_nonstatus_information    1       ; Retain non-status information across program restarts
        notification_interval   0       ; Only send notifications on status change by default.
        is_volatile             0
        check_period            24x7
        normal_check_interval   0.5     # un check cada medio minuto
        retry_check_interval    0.5
        max_check_attempts      4
        notification_period     24x7
        notification_options    w,u,c,r
        contact_groups          admins
        register                0       ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL SERVICE, JUST A TEMPLATE!
}
```

# Servidor de X-Window

## Por defecto

```
Section "ServerLayout"
   Identifier    "XFree86 Configured"
   Screen      0  "Screen0" 0 0
   InputDevice         "Mouse0" "CorePointer"
   InputDevice         "Keyboard0" "CoreKeyboard"
EndSection

Section "Files"
   RgbPath    "/usr/X11R6/lib/X11/rgb"
   ModulePath    "/usr/X11R6/lib/modules"
   FontPath    "/usr/X11R6/lib/X11/fonts/misc/"
   FontPath    "/usr/X11R6/lib/X11/fonts/TTF/"
   FontPath    "/usr/X11R6/lib/X11/fonts/Speedo/"
   FontPath    "/usr/X11R6/lib/X11/fonts/Type1/"
   FontPath    "/usr/X11R6/lib/X11/fonts/CID/"
   FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/"
   FontPath    "/usr/X11R6/lib/X11/fonts/100dpi/"
EndSection

Section "Module"
   Load  "glx"
   Load  "extmod"
   Load  "dri"
   Load  "record"
   Load  "xtrap"
   Load  "dbe"
```

```
    Load  "type1"
    Load  "speedo"
EndSection
Section "InputDevice"
    Identifier  "Keyboard0"
    Driver      "keyboard"
EndSection

Section "InputDevice"
    Identifier  "Mouse0"
    Driver      "mouse"
    Option      "Protocol" "auto"
    Option      "Device" "/dev/mouse"
EndSection

Section "Monitor"
    Identifier   "Monitor0"
    VendorName   "Monitor Vendor"
    ModelName    "Monitor Model"
EndSection

Section "Device"
        ### Available Driver options are:-
        ### Values: <i>: integer, <f>: float, <bool>: "True"/"False",
        ### <string>: "String", <freq>: "<f> Hz/kHz/MHz"
        ### [arg]: arg optional
        #Option     "ShadowFB"              # [<bool>]
        #Option     "DefaultRefresh"        # [<bool>]
        #Option     "VBEBigEndian"          # [<bool>]
    Identifier  "Card0"
    Driver      "vesa"
    VendorName  "nVidia Corporation"
    BoardName   "Unknown Board"
    BusID       "PCI:1:0:0"
```

```
EndSection

Section "Screen"
    Identifier "Screen0"
    Device     "Card0"
    Monitor    "Monitor0"
    SubSection "Display"
        Viewport   0 0
        Depth      1
    EndSubSection
    SubSection "Display"
        Viewport   0 0
        Depth      4
    EndSubSection
    SubSection "Display"
        Viewport   0 0
        Depth      8
    EndSubSection
    SubSection "Display"
        Viewport   0 0
        Depth      15
    EndSubSection
    SubSection "Display"
        Viewport   0 0
        Depth      16
    EndSubSection
    SubSection "Display"
        Viewport   0 0
        Depth      24
    EndSubSection
EndSection
```

## Corregido

```
Section "ServerLayout"
   Identifier    "XFree86 Configured"
   Screen      0  "Screen0" 0 0
   InputDevice         "Keyboard0" "CoreKeyboard"
EndSection

Section "Files"
   RgbPath   "/usr/X11R6/lib/X11/rgb"
   ModulePath   "/usr/X11R6/lib/modules"
   FontPath   "/usr/X11R6/lib/X11/fonts/misc/"
   FontPath   "/usr/X11R6/lib/X11/fonts/TTF/"
   FontPath   "/usr/X11R6/lib/X11/fonts/Speedo/"
   FontPath   "/usr/X11R6/lib/X11/fonts/Type1/"
   FontPath   "/usr/X11R6/lib/X11/fonts/CID/"
   FontPath   "/usr/X11R6/lib/X11/fonts/75dpi/"
   FontPath   "/usr/X11R6/lib/X11/fonts/100dpi/"
EndSection

Section "Module"
   Load  "glx"
   Load  "extmod"
   Load  "dri"
   Load  "record"
   Load  "xtrap"
   Load  "dbe"
   Load  "type1"
   Load  "speedo"
EndSection
```

```
Section "InputDevice"
    Identifier  "Keyboard0"
    Driver      "keyboard"
EndSection

Section "Monitor"
    Identifier   "Monitor0"
    VendorName   "Monitor Vendor"
    ModelName    "Monitor Model"
    HorizSync 31.5-50.0
    VertRefresh  56.0 - 65.0
    modeline "1280x800"   83.50  1280 1352 1480 1680  800 803 809 831 -hsync +vsync
EndSection

Section "Device"
        ### Available Driver options are:-
        ### Values: <i>: integer, <f>: float, <bool>: "True"/"False",
        ### <string>: "String", <freq>: "<f> Hz/kHz/MHz"
        ### [arg]: arg optional
        #Option     "ShadowFB"              # [<bool>]
        #Option     "DefaultRefresh"        # [<bool>]
        #Option     "VBEBigEndian"          # [<bool>]
    Identifier  "Card0"
    Driver      "vesa"
    VendorName  "nVidia Corporation"
    BoardName   "Unknown Board"
    BusID       "PCI:1:0:0"
EndSection

Section "Screen"
    Identifier "Screen0"
    Device     "Card0"
    Monitor    "Monitor0"
    DefaultDepth 16
```

```
    SubSection "Display"
        Viewport   0 0
        Depth      1
    EndSubSection
    SubSection "Display"
        Viewport   0 0
        Depth      4
    EndSubSection
    SubSection "Display"
        Viewport   0 0
        Depth      8
    EndSubSection
    SubSection "Display"
        Viewport   0 0
        Depth      15
    EndSubSection
    SubSection "Display"
        Viewport   0 0
        Depth      16
        Modes      "1280x800"
    EndSubSection
    SubSection "Display"
        Viewport   0 0
        Depth      24
    EndSubSection
EndSection
```

# Anexo C - Errores de configuración/ compilación/instalación

En este anexo presentamos algunos de los errores, suficientemente importantes como para comentarlos, con los que nos hemos encontrado durante el desarrollo del proyecto. Los describimos aquí junto con la solución que hemos encontrado.

## Al compilar por primera vez el compilador gcc

/usr/include/linux/errno.h:4:23: error fatal: asm/errno.h: No existe el fichero o el directorio compilación terminada.
make[3]: *** [_muldi3.o] Error 1
make[3]: se sale del directorio «/home/pfc/Descargas/gcc-4.6.0/i686-pc-linux-gnu/libgcc»
make[2]: *** [all-stage1-target-libgcc] Error 2
make[2]: se sale del directorio «/home/pfc/Descargas/gcc-4.6.0»
make[1]: *** [stage1-bubble] Error 2
make[1]: se sale del directorio «/home/pfc/Descargas/gcc-4.6.0»
make: *** [all] Error 2

No encuentra el archivo /usr/include/linux/errno.h, según he podido encontrar, es un bug de Ubuntu 11.04, hay que instalar gcc-multilib, que crea un symbolic link que restaura /usr/include/linux/ al mismo estado que en Ubuntu 10.10:
sudo apt-get install gcc-multilib;

No se sabe porque pasa esto exactamente, pero se cree que es por algo relacionado con

los compiladores intel y/o los procesadores de 64 bits.

## Al compilar el paquete util-linux

make[2]: se ingresa al directorio «/home/pfc/Descargas/util-linux.ubuntu/mount»
CC mount-mount.o
In file included from mount.c:39:0:
loop.h:40:2: error: nombre de tipo ‘__kernel_dev_t’ desconocido
loop.h:42:2: error: nombre de tipo ‘__kernel_dev_t’ desconocido
make[2]: *** [mount-mount.o] Error 1
make[2]: se sale del directorio «/home/pfc/Descargas/util-linux.ubuntu/mount»
make[1]: *** [all-recursive] Error 1
make[1]: se sale del directorio «/home/pfc/Descargas/util-linux.ubuntu»
make: *** [all] Error 2

No reconoce la variable “__kernel_dev_t” que inicializa en la línea “#define my_dev_t __kernel_dev_t” del fichero ~/Descargas/util-linux.ubuntu/mount/loop.h. Hay sustituir esa línea, por esta otra: “#define my_dev_t dev_t”, porque en el fichero donde está declarada (/usr/src/linux-headers-2.6.38-8-generic/include/linux/types.h) le pone ese nombre a la variable: “typedef __kernel_dev_t **dev_t**;”

## Al compilar X-Window(1)

fbdevhw.c: En la función ‘calculateFbmem_len’:
fbdevhw.c:548:56: error: ‘PAGE_MASK’ no se declaró aquí (primer uso en esta función)
fbdevhw.c:548:56: nota: cada identificador sin declarar se reporta sólo una vez para cada función en el que aparece
fbdevhw.c: En la función ‘fbdevHWMapVidmem’:

fbdevhw.c:575:77: error: ‘PAGE_MASK’ no se declaró aquí (primer uso en esta función)
fbdevhw.c: En la función ‘fbdevHWMapMMIO’:
fbdevhw.c:620:54: error: ‘PAGE_MASK’ no se declaró aquí (primer uso en esta función)
fbdevhw.c: En la función ‘fbdevHWUnmapMMIO’:
fbdevhw.c:644:13: error: ‘PAGE_MASK’ no se declaró aquí (primer uso en esta función)
make[7]: *** [fbdevhw.o] Error 1
make[6]: *** [all] Error 2
make[5]: *** [all] Error 2
make[4]: *** [hw] Error 2
make[3]: *** [all] Error 2
make[2]: *** [all] Error 2
make[1]: *** [World] Error 2
make: *** [World] Error 2

La variable PAGE_MASK está en el archivo /home/pfc/usr/include/asm/page_types.h, que está incluido en el archivo /home/pfc/usr/include/asm/page.h, que es el que incluye el archivo fbdevhw.c, pero no tiene esa variable declarada, porque la estructura de esos ficheros se ha cambiado, y actualmente el fichero page.h solo incluye page_types.h si no están definidas las cabeceras del kernel.

Para solucionarlo, simplemente añadimos en el fichero /home/pfc/Descargas/xfree86_build/programs/Xserver/hw/xfree86/fbdevhw/fbdevhw.c la línea:
#include "asm/page_types.h"
justo debajo de:
#include "asm/page.h"  /* #define for PAGE_* */

## Al compilar X-Window(2)

vm86.c: En la función ‘Vm86DoInterrupt’:
vm86.c:272:27: error: ‘IF_MASK’ no se declaró aquí (primer uso en esta función)
vm86.c:272:27: nota: cada identificador sin declarar se reporta sólo una vez para cada función en el que aparece
vm86.c:272:37: error: ‘IOPL_MASK’ no se declaró aquí (primer uso en esta función)
vm86.c: En la función ‘vm86_emulate’:
vm86.c:388:50: aviso: se define la variable ‘pref_67’ pero no se usa [-Wunused-but-set-variable]
vm86.c:388:9: aviso: se define la variable ‘pref_seg’ pero no se usa [-Wunused-but-set-variable]
make[7]: *** [vm86.o] Error 1
make[6]: *** [all] Error 2
make[5]: *** [all] Error 2
make[4]: *** [hw] Error 2
make[3]: *** [all] Error 2
make[2]: *** [all] Error 2
make[1]: *** [World] Error 2
make: *** [World] Error 2

Se trata de un bug del paquete, para arreglarlo abrimos el archivo /home/pfc/Descargas/xfree86_build/programs/Xserver/hw/tinyx/vesa/vm86.h, y debajo de,
#include “os.h”
#endif

añadimos:
#ifndef IF_MASK
#define IF_MASK X86_EFLAGS_IF
#endif
#ifndef IOPL_MASK
#define IOPL_MASK X86_EFLAGS_IOPL

#endif

## Al compilar X-Window(3)

xcursorgen.c: En la función ‘load_image’:
xcursorgen.c:185:7: error: puntero deferenciado a tipo de dato incompleto
make[4]: *** [xcursorgen.o] Error 1
make[3]: *** [all] Error 2
make[2]: *** [all] Error 2
make[1]: *** [World] Error 2
make: *** [World] Error 2

Desde la versión de libpng-1.5.0 el struct png_info (que es lo que retorna la función png_create_info_struct) ya no se puede acceder desde las aplicaciones. Tenemos que cambiar la línea 185 de /home/pfc/Descargas/xfree86_build/programs/xcursorgen/xcursorgen.c:
 if (setjmp (png->jmpbuf))

por esta otra:
 if (setjmp (png_jmpbuf(png)))

Tal como se explica en el fichero png.h

## Al compilar X-Window(4)

../../../exports/bin/xcursorgen: error while loading shared libraries: libpng15.so.15: cannot open shared object file: No such file or directory
make[5]: *** [X_cursor] Error 127
make[4]: *** [all] Error 2
make[3]: *** [all] Error 2

make[2]: *** [all] Error 2
make[1]: *** [World] Error 2
make: *** [World] Error 2

No encuentra las bibliotecas de png. Si miramos el archivo World.log que ha creado al hacer la compilación, al final vemos esta línea:
LD_LIBRARY_PATH=../../../exports/lib ../../../exports/bin/xcursorgen X_cursor.cfg X_cursor

También se puede ver lo ha hecho en:
make[5]: se ingresa al directorio «/home/pfc/Descargas/xfree86_build/programs/xcursorgen/redglass»

Por lo que abrimos el Makefile de ese directorio, vemos que utiliza la variable LD_LIBRARY_PATH. Ahora bastaría con añadirle la ruta donde está la biblioteca libpng15.so.15, pero al hacer el make, primero hace un clean, por lo que este fichero Makefile es borrado, junto con todas las bibliotecas que utiliza, y por lo tanto, no nos serviría.
Miramos en el archivo World.log de nuevo para ver donde utiliza el directorio de las bibliotecas que utiliza en el make(.../exports/lib), y vemos que en el momento en el que hace el Imakefile de .../config/util ya están todas las bibliotecas en su sitio, así que abrimos el archivo /home/pfc/Descargas/xfree86_build/config/util/Imakefile y en la línea siguiente donde pone:
all:: xmkmf mergelib $(CCMDEP_PROG) $(GCCMDEP_PROG) $(PROGRAMS)

añadimos:

```
        ln -s /home/pfc/usr/lib/libpng15.so.15   \
/home/pfc/Descargas/xfree86_build/exports/lib/
```

Es muy importante que los espacios que hay delante de “ln” sean de apretar una vez la tecla tabulador, si no el make fallará.