Creando un CDROM autoejecutable con ISOLINUX

Sergio González González
HispaLinux, España

sergio.gonzalez@hispalinux.es

Antes de pasar el control al proceso INIT, hay que realizar una serie de tareas. Este documento trata de explicarlas.

Introducción

Normalmente, las distribuciones creadas a partir del proyecto Metadistros (http://metadistros.hispalinux.es/) arrancarán desde un CDROM. El proceso de arranque tiene varias etapas para su consecución, siendo el objetivo de este documento aquella que va desde que la BIOS da el control al CDROM hasta que el "CDROM" da el control al proceso INIT[1]. Para ello deberemos crear un CD capaz de arrancar un sistema (utilizaremos ISOLINUX (http://syslinux.zytor.com/iso.php)) y que pase el control al proceso INIT en un determinado momento.

Importante: Tanto el desarrollo como el presente documento sólo ha tenido en cuenta las necesidades (en cuanto a archivos) para lograr pasar el control al proceso INIT en un determinado momento. Las necesidades particulares de cada metadistribución han de ser cubiertas por el autor de la misma, esto se refleja en la adicción de nuevos archivos, scripts, etc.

Nota: KNOPPIX (http://www.knoppix.org) y Gentoo (http://www.gentoo.org) han sido dos de las distribuciones en las cuales se ha basado este desarrollo.

ISOLINUX

Para poder crear un CDROM autoejecutable, en metadistros hacemos uso de ISOLINUX (http://syslinux.zytor.com/iso.php). Veamos un ejemplo de como crear un CD de estas características con esta aplicación. Las siguientes secciones mostrarán los archivos y configuraciones utilizadas en metadistros para crear un CD autoejecutable con ISOLINUX.

En el ejemplo trabajaremos bajo un directorio vacío: `/mnt/metadistros/master`. Este será la raíz de nuestro CD. Antes de proceder, creamos el directorio `isolinux`, que almacenará los archivos necesarios para que ISOLINUX cumpla con su misión. Los archivos que hemos de incluir en el directorio `isolinux` se listan en las siguientes secciones:

isolinux.bin

Imagen de arranque para los CDs autoejecutables que utilizan “El Torito”. Este archivo lo provee syslinux (http://syslinux.zytor.com/) [2], copielo al directorio `isolinux` del CDROM.

isolinux.cat

Este archivo se crea a la hora de hacer la imagen ISO con **mkisofs**, por lo que no ha de preocuparnos. Únicamente hemos de tener cuidado de no utilizar ningún archivo con ese nombre bajo el directorio `/mnt/metadistros/master/isolinux`. Este archivo se utiliza a la hora de crear un CD autoejecutable "El Torito", como es nuestro caso.

isolinux.cfg

Archivo de configuración de ISOLINUX. Toda la información necesaria para configurar ISOLINUX está disponible en su página (http://syslinux.zytor.com/faq.php#config), por lo que no me extenderé mucho en su explicación.

El cuadro siguiente muestra el archivo de configuración utilizado en metadistros:

```
default sbm-usuario

timeout 300
prompt 1
kbdmap spanish.kbd
display isolinux.msg
f1 isolinux.msg
f2 f2.msg
f3 f3.msg

label sbm-usuario
 kernel vmlinuz.usuario
 append lang=es init=/linuxrc apm=power-off hda=scsi hdb=scsi \
               hdc=scsi hdd=scsi hde=scsi hdf=scsi hdg=scsi hdh=scsi \
               vga=785 initrd=initrd.gz quiet nomce
label sbm-servidor
 kernel vmlinuz.servidor
 append lang=es init=/linuxrc verbose apm=power-off hda=scsi \
               hdb=scsi hdc=scsi hdd=scsi hde=scsi hdf=scsi hdg=scsi \
               hdh=scsi vga=normal initrd=initrd.gz quiet nomce
label memtest
 kernel memtest
```

Las opciones listadas en el cuadro anterior serán brevemente explicadas a continuación:

- "default": opciones por defecto para el arranque por defecto, si no se pasa ningún parámetro en el arranque, se aplicarán las opciones aquí listadas.
- "timeout": tiempo de espera antes de comenzar el arranque desde el CD (en unidades de 1/10 s).
- "prompt": si el valor está a *1*, el arranque mostrará un prompt (boot:). Si el valor está a *0*, el arranque sólo mostrará el prompt si se pulsan las teclas **Alt** o **Shift**.
- "kbdmap": carga el mapa de teclado indicado.
- "display": muestra el archivo indicado antes del prompt.
- "f1, f2 y f3": muestra los archivos indicados al pulsar las teclas de función en el prompt.
- "label": etiqueta que identifica una determinada imagen y sus opciones asociadas (sobreescribirán la opciones por defecto).
- "kernel": imagen a utilizar para el arranque.
- "append": opciones pasadas a la imagen definida en la etiqueta de la que cuelga.

`isolinux.msg`

En nuestro caso, este archivo contiene el "saludo" inicial de la *metadistribución*[3], y tiene un aspecto similar a:

```
^O17^L^Xlogo.16

Sistema Base Metadistros 0.1         http://metadistros.hispalinux.es/
Presione las teclas F2 y F3 para ver la ayuda.
```

Si se fija, al comienzo del archivo, hay una serie de códigos especiales que ISOLINUX se encarga de interpretar[4]. Veamos que hacen estos códigos:

- *^O17* (**Ctrl+o**17): Establece como color de fondo (primer número) un azul oscuro y de primer plano (segundo número) un gris claro.
- *^L* (**Ctrl+l**): Borra la pantalla y establece los colores definidos anteriormente.
- *^Xlogo.16* (**Ctrl+x**logo.16): muestra el gráfico incluido en el archivo logo.16.

> **Nota:** Si utiliza **vi(m)** como editor, para teclear los caracteres de control anteriormente listados, ha de teclearlos en modo *visual*. De forma que, si queremos añadir el código *^L* (**Ctrl+L**) hemos de hacer lo siguiente: entramos en modo *inserción*, si no lo estábamos ya. Pulsamos **Ctrl+v** para entrar en modo *visual* y seguidamente pulsamos **Ctrl+l**. Una vez realizado esto, ya tendremos correctamente tecleado el código de control.

`logo.16`

Imagen en formato LSS16. Syslinux provee un script (**ppmtolss16**) en perl para producir este tipo de imágenes. Estas imágenes se muestran en una resolución de 640x480 pixels y 16 bits de profundidad de color.

Cuando se entra en modo gráfico, los colores definidos con los códigos antes comentados (por ejemplo: ^014) se tratan de forma distinta: el color de fondo se ignora, y los colores de primer plano se especifican en la imagen (16 en total). Esto se especifica a la hora de crear la imagen en formato LSS16 con **ppmtolss16**.

Veamos como se crea una imagen de este tipo con **ppmtolss16**, pero antes echemos un vistazo a la cabecera del script en cuestión:

```
##
## ppmtolss16
##
## Convert a "raw" PPM file with max 16 colors to a simple RLE-based format:
##
## uint32 0x1413f33d    ; magic (littleendian)
## uint16 xsize         ; littleendian
## uint16 ysize         ; littleendian
## 16 x uint8 r,g,b     ; color map, in 6-bit format (each byte is 0..63)
##
## Then, a sequence of nybbles:
##
## N    ... if N is != previous pixel, one pixel of color N
## ... otherwise run sequence follows ...
## M    ... if M > 0 then run length is M+1
## ... otherwise run sequence is encoded in two nybbles,
##     littleendian, +17
##
## The nybble sequences are on a per-row basis; runs may not extend
## across rows and odd-nybble rows are zero-padded.
##
## At the start of row, the "previous pixel" is assumed to be zero.
##
## BUG: This program does not handle comments in the header, nor
## "plain" ppm format.
##
## Usage:
##
##      ppmtorle16 [#rrggbb=i ...] < input.ppm > output.rle
##
## Command line options of the form #rrggbb=i indicate that
## the color #rrggbb (hex) should be assigned index i (decimal)
##
```

Como podemos observar, esta breve explicación nos da una idea de como funciona este script.

Antes de poder obtener la imagen en formato LSS16, hemos de seleccionar la imagen a utilizar, esta ha de estar en formato PPM, con un máximo de 16 colores. Puedes utilizar The Gimp! (http://www.gimp.org/) para obtenerlo a partir de una imagen, o ImageMagick (http://www.imagemagick.org/) de la siguiente forma:

```
$ convert -depth 8 imagen.png imagen.ppm
```

Una vez tenemos la imagen en formato PPM, procedemos a transformarla a formato LSS16, para ello, tecleamos:

```
$ ppmtolss16 < imagen.ppm > logo.16
```

Hecho esto, copiamos el archivo `logo.16` al directorio `isolinux`.

> **Nota:** En este apartado es necesario profundizar más, por favor añada lo que crea conveniente.

`f2.msg` y `f3.msg`

Los archivos de esta sección son los encargados de mostrar la ayuda al usuario en el arranque. Normalmente tienen las distintas opciones que se le pueden pasar al prompt en el arranque. Como se ha definido en el archivo de configuración (la sección de nombre *`isolinux.cfg`*), a estos archivos se accede pulsando las teclas **F2** y **F3**. Su contenido es el siguiente:

Archivo `f2.msg`

```
^Y^O0f^L                       -  Basic boot options -

    ||   At the prompt type "sbm-usuario option", "memtest" for RAM     ||
    ||         test or just hit enter if no options are needed.         ||
    ||                                                                  ||
    ||   [ F1 to redisplay boot graphic || F3 for additional options ]  ||
       -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -

  - - - - - - -  - - - - - -
| lang|keymap=es|de|be|bg    |
|            =ch|cn|cs|cz    |
|            =da|dk|fi|fr     \       specify language and/or keymap
|            =it|ja|nl|pl     / (default settings: lang=es and keymap=es )
|            =ru|sk|tr|tw    |
|            =uk              |
  - - - - - - -  - - - - - -

interactive    -->       Interactive setup for experts
verbose    ---->       Show information at boot up progress
showdebug    --->       Show debug information
vga=normal  -->        No-framebuffer mode

install    -->         Install mode (no live CD)
```

Archivo `f3.msg`

```
^Y^O0f^L                     -  Additional boot options -

    ||   At the prompt type "sbm-usuario option", "memtest" for RAM     ||
    ||         test or just hit enter if no options are needed.         ||
```

```
    ||   [ F1 to redisplay boot graphic || F2 for basic options ]       ||

  - - - - - - - -  - - - - -
| ip=192.168.1.1           |
| broadcast=192.168.1.255   \              Network configuration
| netmask=255.255.255.0     /  (Set this variables for personal configuration)
| gateway=192.168.1.1      |
  - - - - - - - -  - - - - -

ramsize=1000000  >>>  Default maximum size of dynamic ramdisk in kilobytes

  -----------
| noscsi      >>>>>>>>>>>  not scan for scsi devices
| dofirewire  >  modprobes firewire modules in initrd (for firewire cdroms,etc)
| nousb       >>>>>  disables usb module load from initrd, disables hotplug
| doataraid   >>>>>>  loads ide raid modules from initrd
| dopcmcia    >>>>>>>>>>>  starts pcmcia service
| cdcache     >>>  Cache the entire runtime portion of cd in ram, This
  -----------      uses a lot of RAM , but allows you to umount /mnt/cdrom
                   and mount another cdrom.
```

Al igual que en la la sección de nombre *`isolinux.msg`*, los archivos de esta sección también poseen caracteres de control. Cómo en ambos archivos son los mismos códigos, los explicaré una vez:

- ^*Y* (**Ctrl+Y**): pasamos a modo texto, si estábamos en modo gráfico.
- ^*O* (**Ctrl+O**0f): establecemos los colores negro y blanco para el fondo y el primer plano de la pantalla, respectivamente.
- ^*L* (**Ctrl+L**): borra la pantalla y establece los colores definidos.

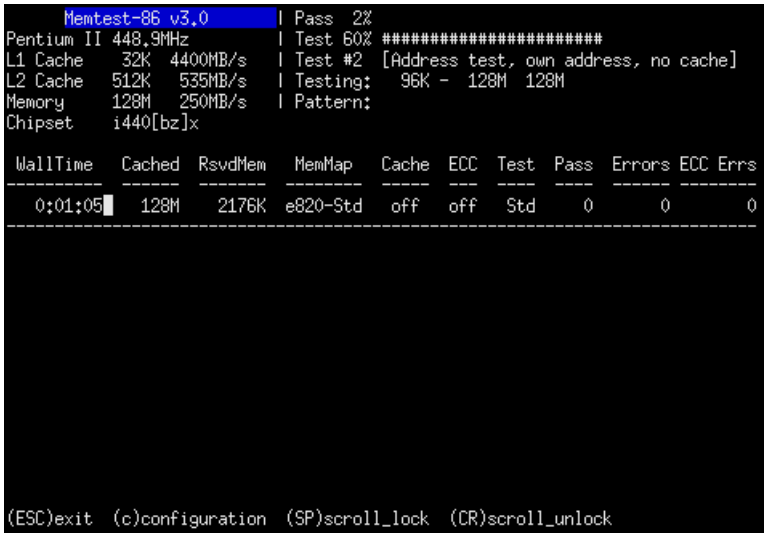
`memtest`

Memtest86 (http://www.memtest86.com/) es un programa de diagnóstico para la memoria RAM. La imagen necesaria para arrancar Memtest86 en el arranque está incluida en el archivo de distribución del programa (que puede encontrar aquí (http://www.memtest86.com/#download0)), por lo que lo copiaremos al directorio `/mnt/metadistros/master/isolinux`.

Para poder utilizar este programa de diagnóstico, en el prompt de arranque teclearemos:

```
boot: memtest
```

Una vez hecho esto, Memtest86 comenzará a ejecutarse:



Captura de pantalla de Memtest86

spanish.kbd

Mapa de teclado utilizado en el prompt de arranque. El siguiente ejemplo creará [5] el mapa de teclado para usuarios españoles. Necesitamos el programa **keytab-lilo** [6] y la definición del teclado español[7], en este caso. Una vez tenemos todo esto, ejecutaremos:

```
$ keytab-lilo /usr/share/keymaps/i386/qwerty/es.kmap.gz > spanish.kbd
Loading /usr/share/keymaps/i386/qwerty/us.kmap.gz
Loading /usr/share/keymaps/i386/qwerty/es.kmap.gz
$
```

Una vez finalice su ejecución, tendremos un nuevo archivo, `spanish.kbd`, el cual copiaremos al directorio `/mnt/metadistros/master/isolinux`. De esta forma, y como vimos en la la sección de nombre *`isolinux.cfg`*, al arrancar el sistema desde el CDROM, tendremos nuestro mapa de teclado en español.

vmlinuz.usuario

Núcleo para nuestra distribución. En este caso se ha utilizado el núcleo destinado a sistemas de escritorio que provee Metadistros. Puede descargarlo de cualquier mirror disponible.

initrd.gz

En este apartado vamos a crear un archivo initrd, que no es otra cosa que un un disco RAM que es inicializado (es decir, cargado) por el gestor de arranque antes de cargar e iniciar el núcleo del sistema (como dice la página del manual **man initrd**).

En el siguiente ejemplo crearemos un archivo de 5 megabytes de tamaño, en cuyo interior se podrán colocar los scripts, programas, módulos y dispositivos necesarios para arrancar el sistema. La forma de crear este archivo es con el comando **dd**, como vemos a continuación:

```
$ dd if=/dev/zero of=initrd bs=5000k count=1
1+0 registros leídos
1+0 registros escritos
5120000 bytes transferred in 0,238769 seconds (21443318 bytes/sec)
```

Una vez ejecutado el anterior comando, tendremos disponible un archivo llamado `initrd`, el cual ya podremos formatear con el sistema de archivos ext2:

```
$ mke2fs -v -N 7000 initrd
mke2fs 1.33 (21-Apr-2003)
initrd is not a block special device.
Proceed anyway? (y,n)
```

A esta pregunta respondemos afirmativamente, con lo que nos creará un sistema de ficheros ext2 en el interior del archivo `initrd`:

```
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
7000 inodes, 5000 blocks
250 blocks (5.00%) reserved for the super user
First data block=1
1 block group
8192 blocks per group, 8192 fragments per group
7000 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 24 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

> **Nota:** Hemos pasado el parámetro *`-N 7000`* a **mke2fs**, que sobreescribe el número de inodos que se crearán por defecto, al número especificado en el parámetro. Esto es necesario debido a que crea un número muy bajo de inodos por defecto, que al utilizar archivos de tamaño muy pequeño y en gran cantidad, se terminan en seguida.

Aviso

El número de inodos especificados anteriormente es orientativo y puede ser necesario variarlo dependiendo del tamaño del archivo `initrd`. Dicho número, en este caso, no pude ser superior a 7200 aprox., ya que si se eleva por encima de dicha cifra, el sistema no arrancará.

Ahora sólo nos queda afinar el sistema de ficheros que hemos creado e incorporarle la información necesaria en su interior. En el afinado, ampliaremos el número de veces que el sistema puede ser montado sin necesidad de que este sea chequeado. Este número lo ajustaremos alto, ya que montaremos muchas veces el archivo en cuestión:

```
$ tune2fs -c 1000 initrd
tune2fs 1.33 (21-Apr-2003)
Setting maximal mount count to 1000
```

Ahora ya estamos en disposición de montar el archivo que hemos creado, para ello teclearemos, por ejemplo:

```
$ mount -o loop -t ext2 initrd /mnt/initrd
```

Una vez montado, copiaremos los archivos necesarios en su interior (vea la la sección de nombre *Contenido de `initrd`*) y lo comprimiremos (una vez desmontado).

```
$ umount /mnt/initrd
$ gzip -9 initrd
```

Resumiendo

Hasta aquí hemos visto los archivos utilizados en metadistros para ISOLINUX, en la siguiente sección, veremos el contenido del archivo `initrd`.

El listado de archivos que se ve a continuación, son los que hemos repasado en esta sección:

```
$ tree isolinux
isolinux/
|-- f2.msg
|-- f3.msg
|-- initrd
|-- isolinux.bin
|-- isolinux.cat
|-- isolinux.cfg
|-- isolinux.msg
|-- logo.16
|-- memtest
|-- spanish.kbd
‘-- vmlinuz.usuario
```

Contenido de `initrd`

Antes de comenzar con este apartado, veamos la estructura del sistema de archivos que se encuentra contenido en el archivo `initrd`. Aquí sólo se listarán los directorios de dicho sistema de ficheros (que se irán analizando a lo largo de esta sección):

```
$ mount -o loop -t ext2 initrd /mnt/initrd
```

```
$ cd /mnt/initrd
$ tree -d
.
|-- META
|-- cdrom
|-- dev
|   |-- ataraid
|   |-- cciss
|   |-- fd -> /proc/self/fd
|   |-- ida
|   |-- pts
|   `-- rd
|-- etc
|   `-- metadistros
|-- keymaps
|-- mnt
|-- modules
|   |-- ataraid
|   |-- firewire
|   |-- others
|   |-- pcmcia
|   |-- scsi
|   `-- usb
|-- proc
|-- static
`-- tmp -> /var/tmp

23 directories
```

El directorio “raíz” y el directorio `META`

En este apartado tratamos estos dos directorios debido a que en el “raíz” sólo nos falta por ver una serie de enlaces simbólicos que están relacionados con el directorio `META`. Veamos estos enlaces, para ello ejecutamos en el directorio “raíz” del sistema de archivos contenido en `initrd` lo siguiente:

> **Nota:** En el directorio “raíz” está el script `linuxrc` pero este se tratará en la la sección de nombre *`linuxrc`*, por lo que aquí no comentaremos nada acerca de él.

```
$ tree -L 1
.
|-- META
|-- bin -> /META/bin
|-- boot -> /META/boot
|-- cdrom
|-- dev
|-- etc
|-- keymaps
|-- lib -> /META/lib
|-- linuxrc
|-- mnt
```

```
|-- modules
|-- opt -> /META/opt
|-- proc
|-- sbin -> /META/sbin
|-- static
|-- tmp -> /var/tmp
‘-- usr -> /META/usr

10 directories, 7 files
```

Como podemos observar en la captura anterior, casi todos los enlaces simbólicos[8] apuntan a distintos directorios del directorio `META`, excepto el enlace simbólico `tmp` que apunta al directorio `/var/tmp`. Esto se hace para unificar los directorios temporales, ya que han de tener permiso de escritura.

Veamos el contenido del directorio `META` antes de proceder a dar una breve explicación:

```
$ tree META
META/
|-- bin -> /cdrom/META/bin
|-- boot -> /cdrom/META/boot
|-- etc -> /cdrom/META/etc
|-- lib -> /cdrom/META/lib
|-- opt -> /cdrom/META/opt
|-- sbin -> /cdrom/META/sbin
|-- usr -> /cdrom/META/usr
‘-- var -> /cdrom/META/var

0 directories, 8 files
```

Como se puede ver, el directorio `META` sólo contiene enlaces simbólicos al lugar donde se montará la imagen (ya sea cloop o loop) que posee la distribución a ejecutarse. Estos enlaces serán vitales cuando se establezca el PATH genérico de nuestro sistema.

El directorio `modules`

Bajo el directorio `modules` se encuentran los controladores[9] para dispositivos “raid IDE”, “firewire” (también conocido como “ieee1394”), “pcmcia”, “SCSI” y “USB”. Aunque actualmente no esté implementado

> **Nota:** Sería interesante añadir los módulos necesarios para dar soporte a EVMS (Enterprise Volume Management System). De esta forma, se podría ampliar el número de dispositivos y sistemas de archivos sobre los cuales poder arrancar el sistema. Continuando con esta tónica, también podría analizarse si es interesante añadir soporte para RAID por software.

Se puede observar que el módulo “cloop” también está en este directorio, el cual es necesario para montar imágenes comprimidas en el loopback.

Por último, el directorio `others` se usa en aquellos casos en los que el usuario requiere cargar más módulos de los disponibles por defecto (esto es posible gracias al modo interactivo de arranque).

Nota: Como la imagen `initrd.gz` final es muy grande para que entre en un disquete de 1.44 Mbytes, se pueden eliminar los módulos[10] de esta e incorporarlos en el arranque gracias al modo interactivo. Esto puede ser útil para aquellos equipos que no puedan arrancar desde CDROM.

```
$ tree modules
modules/
|-- ataraid
|   |-- ataraid.o
|   |-- hptraid.o
|   `-- pdcraid.o
|-- cloop.o
|-- firewire
|   |-- eth1394.o
|   |-- ieee1394.o
|   |-- ohci1394.o
|   `-- sbp2.o
|-- others
|-- pcmcia
|   |-- ds.o
|   |-- i82365.o
|   |-- ide-cs.o
|   `-- pcmcia_core.o
|-- scsi
|   |-- 3w-xxxx.o
|   |-- BusLogic.o
|   |-- NCR53c406a.o
|   |-- a100u2w.o
|   |-- advansys.o
|   |-- aha152x.o
|   |-- aha1542.o
|   |-- aha1740.o
|   |-- aic7xxx.o
|   |-- atp870u.o
|   |-- dtc.o
|   |-- eata.o
|   |-- fdomain.o
|   |-- gdth.o
|   |-- initio.o
|   |-- megaraid.o
|   |-- ncr53c8xx.o
|   |-- pas16.o
|   |-- pci2000.o
|   |-- pci2220i.o
|   |-- psi240i.o
|   |-- qlogicfas.o
|   |-- qlogicfc.o
|   |-- qlogicisp.o
|   |-- seagate.o
|   |-- t128.o
|   |-- tmscsim.o
|   |-- u14-34f.o
|   |-- ultrastor.o
```

```
|   ‘-- wd7000.o
‘-- usb
    |-- ehci-hcd.o
    |-- hid.o
    |-- uhci.o
    |-- usb-ohci.o
    |-- usb-storage.o
    ‘-- usbcore.o

6 directories, 48 files
```

El directorio `keymaps`

Directorio que posee mapas de teclados de distintos países. El mapa de teclado se cargará en aquellos momentos en los que el usuario ha de teclear algo, bien sea por arrancar en modo interactivo bien por no haber encontrado el directorio `META` o la imagen con el sistema a cargar. En este último caso se cargará una shell reducida, que se detallará en la
la sección de nombre *El directorio `static` y la herramienta BusyBox*, desde la cual el usuario podrá hacer uso de distintas herramientas.

> **Nota:** Estos mapas de teclado no serán necesario una vez tengamos cargado el sistema de la imagen (c)loop, ya que dicho sistema ya traerá distintos mapas de teclados. La utilidad de este directorio es proporcionar comodidad al usuario final en determinadas ocasiones. Si no poseemos mucho espacio[11] para la imagen de arranque, por tener que arrancar desde disquete, por ejemplo, podremos eliminarlo sin problemas.

```
$ tree keymaps
keymaps/
|-- 1.map
|-- 10.map
|-- 11.map
|-- 12.map
|-- 13.map
|-- 14.map
|-- 15.map
|-- 16.map
|-- 17.map
|-- 18.map
|-- 19.map
|-- 2.map
|-- 20.map
|-- 21.map
|-- 22.map
|-- 23.map
|-- 24.map
|-- 25.map
|-- 26.map
|-- 27.map
|-- 28.map
```

```
|-- 29.map
|-- 3.map
|-- 30.map
|-- 31.map
|-- 32.map
|-- 33.map
|-- 34.map
|-- 35.map
|-- 36.map
|-- 37.map
|-- 38.map
|-- 39.map
|-- 4.map
|-- 40.map
|-- 41.map
|-- 42.map
|-- 5.map
|-- 6.map
|-- 7.map
|-- 8.map
|-- 9.map
|-- azerty.map
|-- be.map
|-- bg.map
|-- br-a.map
|-- br-l.map
|-- by.map
|-- cf.map
|-- croat.map
|-- cz.map
|-- de.map
|-- dk.map
|-- dvorak.map
|-- es.map
|-- et.map
|-- fi.map
|-- fr.map
|-- gr.map
|-- hu.map
|-- il.map
|-- is.map
|-- it.map
|-- jp.map
|-- key.lst
|-- la.map
|-- lt.map
|-- mk.map
|-- nl.map
|-- no.map
|-- pl.map
|-- pt.map
|-- ro.map
|-- ru.map
```

```
|-- se.map
|-- sg.map
|-- sk-y.map
|-- sk-z.map
|-- slovene.map
|-- trf.map
|-- trq.map
|-- ua.map
|-- uk.map
|-- us.map
`-- wangbe.map

0 directories, 85 files
```

El directorio dev

Directorio que contiene los distintos dispositivos del sistema. Como el listado de archivos es muy grande, sólo dejaremos el número de archivos (como curiosidad) y el tamaño de este directorio:

```
$ tree dev
dev/
|-- ataraid
|   |-- d0
|   |-- d0p1
|   |-- d0p10

[...]

|-- xdb5
|-- xdb6
|-- xdb7
|-- xdb8
`-- zero

6 directories, 4812 files
$ du -h dev
49K     dev/rd
19K     dev/ida
512     dev/pts
19K     dev/cciss
1,5K    dev/ataraid
130K    dev
```

> **Nota:** Se podría ganar algo de espacio utilizando el sistema de archivos “devfs” (the Linux Device Filesystem). El único problema, es que habría que adaptar todos los scripts para que funcionasen con el mismo.

El directorio `etc`

A continuación se mostrará el contenido de los archivos existentes en el directorio `etc`, excepto los que están bajo el directorio `metadistros`, ya que se verán en la
la sección de nombre *Scripts de arranque - linuxrc*.

El listado de archivos bajo `etc` es:

```
$ tree etc
etc/
|-- auto.mnt
|-- exports
|-- filesystems
|-- fstab
|-- group
|-- ld.so.conf -> /META/etc/ld.so.conf
|-- metadistros
|   |-- functions
|   |-- i18n.conf
|   ‘-- var.conf
|-- mtab
|-- passwd
|-- resolv.conf
‘-- shadow

1 directory, 13 files
```

Veamos pues el contenido de estos archivos:

El archivo `/etc/fstab`

```
/dev/ram0  /           cramfs defaults
proc       /proc       proc   defaults                  0 0
pts        /dev/pts    devpts mode=0622                 0 0
/dev/fd0   /mnt/floppy auto   user,noauto,exec,umask=000 0 0
/dev/cdrom /mnt/cdrom  auto   user,noauto,exec,ro        0 0
```

El archivo `/etc/group`

```
root::0:root
slocate::21:root
nobody::-1:nobody
```

El archivo `/etc/mtab`

```
/dev/root / ext2 rw 0 0
/dev/cdrom /cdrom iso9660 ro 0 0
/dev/cloop /META iso9660 ro 0 0
```

El archivo `/etc/passwd`

```
root:*:0:0:Metadistros Administrator:/root:/bin/bash
nobody:*:-1:-1:Nobody:/:/bin/true
```

El archivo `/etc/resolv.conf`

```
# Insert nameservers here
# nameserver 127.0.0.1
```

El archivo `/etc/shadow`

```
root:*:11312:0:99999:7:::
nobody:*:11312:0:99999:7:::
```

El directorio `static` y la herramienta BusyBox

Busybox (http://www.busybox.net) es una herramienta que provee distintos comandos UNIX en un único ejecutable. Los comandos que han sido compilados para el `initrd` de metadistros se muestran a continuación:

> **Nota:** Busybox se ha alojado en el directorio `static` del `initrd`, y es a ese directorio donde apunta el PATH inicial de los scripts.

```
$ ./static/busybox
BusyBox v0.60.5 (2003.05.11-12:22+0000) multi-call binary

Usage: busybox [function] [arguments]...
   or: [function] [arguments]...

        BusyBox is a multi-call binary that combines many common Unix
        utilities into a single executable.  Most people will create a
        link to busybox for each function they wish to use, and BusyBox
        will act like whatever it was invoked as.

Currently defined functions:
        [, ar, ash, basename, busybox, cat, chgrp, chmod, chown, chroot,
        chvt, clear, cmp, cp, cpio, cut, date, dc, dd, deallocvt, df,
        dirname, dmesg, dos2unix, dpkg, dpkg-deb, du, dumpkmap, dutmp,
        echo, egrep, env, expr, false, fbset, fdflush, find, free, freeramdisk,
        fsck.minix, getopt, grep, gunzip, gzip, halt, head, hostid, hostname,
        id, ifconfig, init, insmod, kill, killall, klogd, length, linuxrc,
        ln, loadacm, loadfont, loadkmap, logger, logname, losetup, ls,
        lsmod, makedevs, md5sum, mkdir, mkfifo, mkfs.minix, mknod, mkswap,
        mktemp, modprobe, more, mount, mt, mv, nc, nslookup, pidof, ping,
```

```
        pivot_root, poweroff, printf, ps, pwd, rdate, readlink, reboot,
        renice, reset, rm, rmdir, rmmod, route, rpm2cpio, sed, setkeycodes,
        sh, sleep, sort, stty, swapoff, swapon, sync, syslogd, tail, tar,
        tee, telnet, test, tftp, time, top, touch, tr, traceroute, true,
        tty, umount, uname, uniq, unix2dos, update, uptime, usleep, uudecode,
        uuencode, vi, watchdog, wc, wget, which, whoami, xargs, yes, zcat
```

Estos comandos son necesarios para la ejecución de los scripts de arranque, ya que proveen, entre otras cosas, la shell desde donde se ejecutan. Los comandos compilados en la shell son:

```
$ ./static/busybox ash
$ help

Built-in commands:
------------------
        . : alias break builtin cd chdir continue eval exec exit export
        false fc hash help jobs let local pwd read readonly return set
        setvar shift times trap true type ulimit umask unalias unset
        wait
```

Hay dos formas de utilizar los comandos que provee busybox, una es pasando como parámetro al ejecutable busybox el comando que se quiere ejecutar:

```
$ ./static/busybox ls
META     boot     dev      keymaps  linuxrc  modules  proc     static   usr
bin      cdrom    etc      lib      mnt      opt      sbin     tmp
```

Otra es haciendo un enlace duro (o físico) desde busybox al nombre de los distintos comandos que esta aplicación provee. De esta forma, si queremos tener el comando ls como tal, crearemos el enlace duro conveniente de la siguiente forma:

```
$ cd static
$ tree
.
‘-- busybox

0 directories, 1 file
$ ln busybox ls
$ tree
.
|-- busybox
‘-- ls

0 directories, 2 files
```

Una vez tenemos el enlace duro ls, al llamarlo, actuará como si hubiésemos pasado el parámetro “ls” a busybox.

```
$ cd static
$ ./ls
busybox  ls
```

Si ahora creamos un enlace duro por cada comando que provee busybox, el directorio `static` nos quedará:

```
$ tree static
static/
|-- [
|-- ash
|-- basename
|-- busybox
|-- cat
|-- chgrp
|-- chmod
|-- chown
|-- chroot
|-- chvt
|-- clear
|-- cmp
|-- cp
|-- cpio
|-- cut
|-- date
|-- dc
|-- dd
|-- deallocvt
|-- df
|-- dirname
|-- dmesg
|-- dos2unix
|-- du
|-- echo
|-- egrep
|-- env
|-- expr
|-- false
|-- fbset
|-- fdflush
|-- find
|-- free
|-- freeramdisk
|-- getopt
|-- grep
|-- gunzip
|-- gzip
|-- halt
|-- head
|-- hostid
|-- hostname
|-- id
|-- ifconfig
|-- init
|-- insmod
|-- kill
|-- killall
```

```
|-- klogd
|-- length
|-- linuxrc
|-- ln
|-- loadacm
|-- loadfont
|-- loadkmap
|-- logger
|-- logname
|-- losetup
|-- ls
|-- lsmod
|-- makedevs
|-- md5sum
|-- mkdir
|-- mkfifo
|-- mkfs.minix
|-- mknod
|-- mkswap
|-- mktemp
|-- modprobe
|-- more
|-- mount
|-- mt
|-- mv
|-- nc
|-- nslookup
|-- pidof
|-- ping
|-- pivot_root
|-- poweroff
|-- printf
|-- ps
|-- pwd
|-- rdate
|-- readlink
|-- reboot
|-- renice
|-- reset
|-- rm
|-- rmdir
|-- rmmod
|-- route
|-- rpm2cpio
|-- sed
|-- setkeycodes
|-- sh
|-- sleep
|-- sort
|-- stty
|-- swapoff
|-- swapon
|-- sync
```

```
|-- syslogd
|-- tail
|-- tar
|-- tee
|-- telnet
|-- test
|-- tftp
|-- time
|-- top
|-- touch
|-- tr
|-- traceroute
|-- true
|-- tty
|-- umount
|-- uname
|-- uniq
|-- unix2dos
|-- update
|-- uptime
|-- usleep
|-- uudecode
|-- uuencode
|-- vi
|-- watchdog
|-- wc
|-- wget
|-- which
|-- whoami
|-- xargs
|-- yes
‘-- zcat

0 directories, 133 files
```

En la Apéndice E se encuentra el archivo de configuración utilizado en metadistros para compilar el ejecutable busybox de modo estático.

Otros directorios: `cdrom`, `mnt` y `proc`

- `cdrom`: directorio donde se montará el CDROM (de todas formas, el CDROM en un futuro se puede corresponder con un “stick USB” o un disco duro externo conectado a través del puerto firewire de nuestro ordenador.
- `mnt`: directorio en el cual se pueden montar distintos medios.
- `proc`: directorio donde se montará el pseudo-sistema de ficheros de información de procesos.

Scripts de arranque - linuxrc

Esta sección no pretende dar una explicación detallada de los scripts de inicio, para eso están las fuentes de los mismos en los apéndices. La función de esta sección es presentar, de forma global, la función de cada uno.

El arranque del CDROM (o del dispositivo que estemos utilizando) es controlado por cuatro archivos: `linuxrc`, `functions`, `i18n.conf` y `var.conf` que se explicarán brevemente a continuación:

`linuxrc`

Script principal del `initrd`, es el que lleva el control del flujo del arranque. En el siguiente gráfico se muestra de forma resumida las acciones lleva a cabo este script hasta darle el control al proceso INIT.

Para más información, lea el Apéndice A.

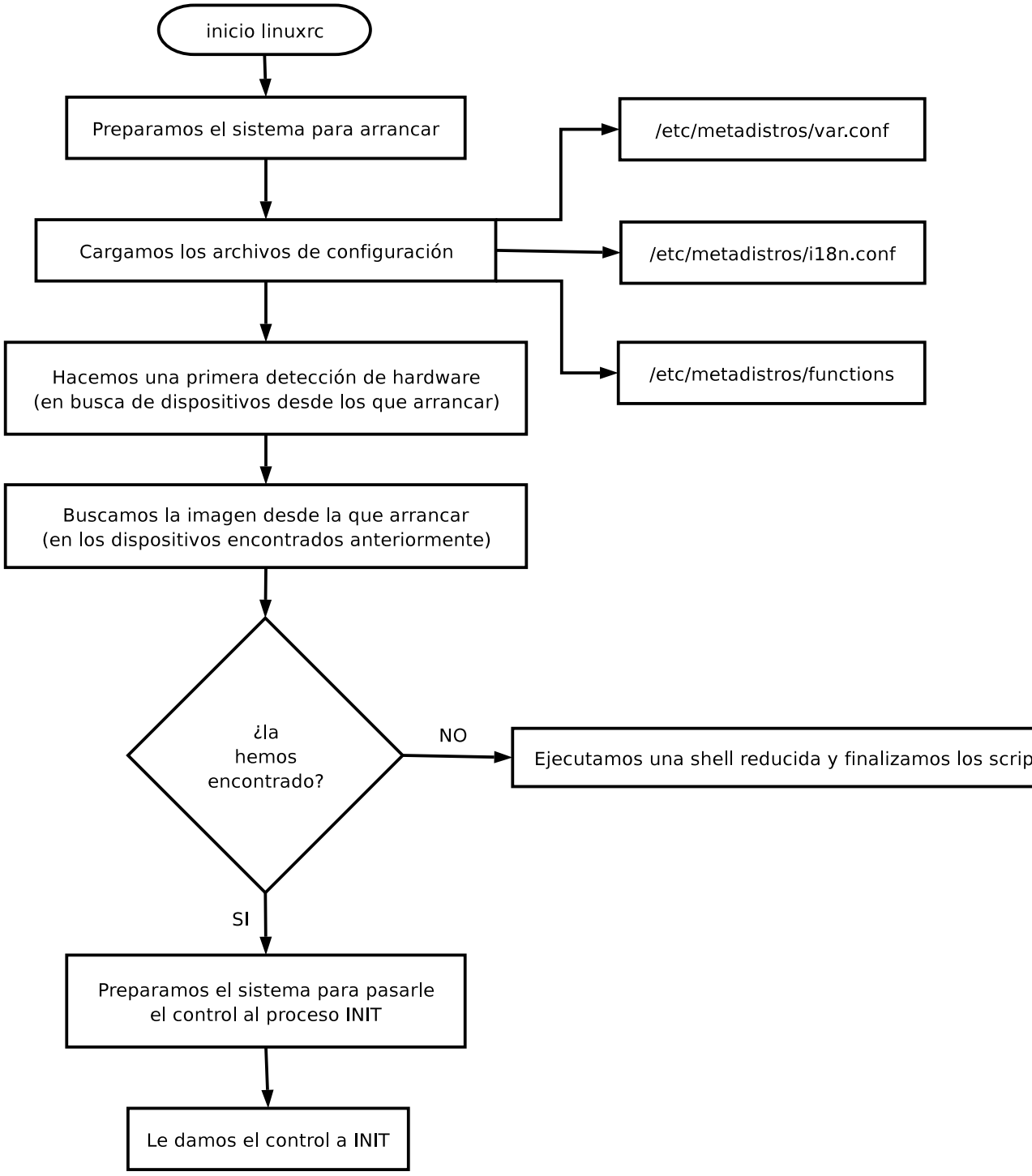


Diagrama de flujo para linuxrc

`functions`

Las definiciones de las funciones utilizadas en el archivo `linuxrc` se encuentran en este archivo y se pueden analizar en el Apéndice B.

`i18n.conf`

Archivo de internacionalización para el arranque. Aquí se configuran las distintas opciones de cada idioma[12], ya sea el mapa de teclado a utilizar o los mensajes que se mostrarán por pantalla en distintas partes del arranque.

En el Apéndice C se encuentra este archivo. Para profundizar en las opciones y mensajes de este, es recomendable su lectura.

`var.conf`

Este es archivo de configuración del arranque y de distintas características de metadistros como el idioma por defecto, la configuración de red, modo de arranque (“verbose”, “debug” o “interactive”), etc.

El Apéndice D muestra el contenido de este archivo.

A. `linuxrc`

```
#!/static/sh
#
# Metadistros boot up script - main program
#
# Based on:
#
#  - KNOPPIX General Startup Script by Klaus Knopper <knoppix@knopper.net>
#  - Gentoo boot up script by Daniel Robbins <drobbins@gentoo.org>
#
# Copyright 2003 Metadistros [ http://metadistros.hispalinux.es/ ]
# Distributed under the GPL


# ============================================================== #
#                       Start script                             #
#
# Don’t allow interrupt signals
trap "" 1 2 3 15

# Set PATH
PATH=/static
export PATH

# Clean input/output
```

```
exec >/dev/console </dev/console 2>&1

umask 022

# Mount /, proc and devpts
mount -o remount,rw /
mount -t proc none /proc
mount -t devpts none /dev/pts


# Enable low latency
echo "1" > /proc/sys/kernel/lowlatency

# Tell kernel where modprobe lives
echo "/static/modprobe" > /proc/sys/kernel/modprobe


# ================================================================ #
#            Load configurations files                             #
#
. /etc/metadistros/var.conf
. /etc/metadistros/i18n.conf
. /etc/metadistros/functions

# Add user defined in var.conf ($USERNAME)
createuser


# ================================================================ #
#                          Welcome message                         #
#

progressbar 51 "${WELCOME}"
if [ "${VERBOSE}" = "yes"  ]; then echo -e "${WHITE}${WELCOME}${NORMAL}\n\n"; fi


# ================================================================ #
#                   First hardware detection                       #
#

# Check for devices, use modules on bootfloppy first
if [ "${INTERACTIVE}" = "yes" ]; then
 # Load keymap
 setkeymap

 # Let the user select interactively
 if [ "${SCSI}" = yes ]
 then
  askmodules scsi     ‘cd /modules/scsi;     echo *.o‘
  SCSI_MODULES="${MODULES}"
 fi
 if [ "${FIREWIRE}" = yes ]
 then
```

```
  askmodules firewire ‘cd /modules/firewire; echo *.o‘
  FIREWIRE_MODULES="${MODULES}"
 fi
 if [ "${ATARAID}" = yes ]
 then
  askmodules ataraid  ‘cd /modules/ataraid;  echo *.o‘
  ATARAID_MODULES="${MODULES}"
 fi
 if [ "${PCMCIA}" = yes ]
 then
  askmodules pcmcia   ‘cd /modules/pcmcia;   echo *.o‘
  PCMCIA_MODULES="${MODULES}"
 fi
 if [ "${USB}" = yes ]
 then
  askmodules usb      ‘cd /modules/usb;      echo *.o‘
  USB_MODULES="${MODULES}"
 fi
fi

# Load selected/default modules
if [ "${FIREWIRE}" = "yes" ] && [ -n "${FIREWIRE_MODULES}" ]
then
 progressbar 52 "${PBMESSAGE1}"
 loadmodules firewire ${FIREWIRE_MODULES}
fi
if [ "${ATARAID}"  = "yes" ] && [ -n "${ATARAID_MODULES}"  ]
then
 progressbar 53 "${PBMESSAGE2}"
 loadmodules ataraid  ${ATARAID_MODULES}
fi
if [ "${PCMCIA}"   = "yes" ] && [ -n "${PCMCIA_MODULES}"   ]
then
 progressbar 54 "${PBMESSAGE3}"
 loadmodules pcmcia   ${PCMCIA_MODULES}
fi
if [ "${USB}"      = "yes" ] && [ -n "${USB_MODULES}"      ]
then
 progressbar 55 "${PBMESSAGE4}"
 loadmodules usb      ${USB_MODULES}
fi
if [ "${SCSI}"     = "yes" ] && [ -n "${SCSI_MODULES}"     ]
then
 progressbar 56 "${PBMESSAGE5}"
 loadmodules scsi     ${SCSI_MODULES}
 FOUND_SCSI=$?
fi
# End of hardware check


# ============================================================= #
#             Check for misc modules in expert mode             #
#
```

```
if [ "${INTERACTIVE}" = "yes" ]; then
 answer=""
 while test "${answer}" != "n" -a "${answer}" != "N"; do
  echo -n "${MESSAGE1}"
  read answer
  case "${answer}" in n*|N*) break; ;; esac
  if mountmodules new; then
   askmodules new 'cd /modules/others; echo *.o'
   test -n "${MODULES}" && loadmodules new ${MODULES}
   umountmodules
  fi
 done
fi
# All interactively requested modules should be loaded now.


# ================================================================ #
#                       Find META directory                        #
#

progressbar 57 "${PBMESSAGE6}"

checkformetadirectory ${FOUND_SCSI}

case $? in
 0)
  FOUND_META="yes"
  ;;
 1)
  FOUND_META="no"
  ;;
esac


# ================================================================ #
#                      load  (c)loop image                         #
#
# Show DEBUG info
[ "${DEBUG}" = "yes" ] && echo "6" > /proc/sys/kernel/printk

# Check for sufficient memory to mount extra ramdisk for /home + /var
progressbar 58 "${PBMESSAGE7}"
checkformemory

# Mount (c)loop image
progressbar 59 "${PBMESSAGE8}"
mountimage

# Disable kernel messages to console again
[ "${DEBUG}" = "no" ] && echo "0" > /proc/sys/kernel/printk


# ================================================================ #
```

```
#            Final test if everything succeeded                    #
#
if [ "${FOUND_META}" != "no" ]; then
 progressbar 60 "${PBMESSAGE9}"

 # Set final PATH
 PATH="/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin"
 export PATH

 # copy library cache
 cat /META/etc/ld.so.cache > /etc/ld.so.cache

 # Verbose mode? -> Enable kernel messages
 if [ "${VERBOSE}" = "yes" ]; then echo "6" > /proc/sys/kernel/printk; fi

 # Debian weirdness
 /bin/cp -a /META/etc/alternatives /etc/ 2>/dev/null

 # From here, we should have all essential commands available.
 hash -r

 # Clean up /
 rm -rf /modules

 # Create common WRITABLE (empty) dirs
 createdirs

 # Now tell kernel where the real modprobe lives
 echo "/sbin/modprobe" > /proc/sys/kernel/modprobe

 # Change root device from /dev/fd0 to /dev/ram0
 echo "0x100" > /proc/sys/kernel/real-root-dev

 # Give control to the init process.
 if [ "${VERBOSE}" = "yes" ]; then echo "${MESSAGE2}"; fi
 rm -f /linuxrc

 # INIT take the control
 exec chroot . /sbin/init </dev/console >/dev/console 2>&1

else
 ( [ "${VERBOSE}" != "yes" ] && [ "${DEBUG}" != "yes" ] ) && echo -e "${CLEAR}"
 echo -e "${MESSAGE3}"
 setkeymap
 echo "6" > /proc/sys/kernel/printk
 PS1="\[^[[36m\]${DISTRO}:\[^[[32;01m\]\w\[^[[0m\]\[^[[36m\]\# "
 export PS1
 # Allow signals
 trap 1 2 3 15
 echo ""

 exec chroot / /static/ash
fi
```

B. functions

```
#!/static/sh
#
# Metadistros boot up script - functions file
#
# Copyright 2003 Metadistros [ http://metadistros.hispalinux.es/ ]
# Distributed under the GPL

# Load configuration and i18n files
. /etc/metadistros/var.conf
. /etc/metadistros/i18n.conf


# ---------------------------------------------------------------------------- #
# Create new user, defined in /etc/metadistros/var.conf

createuser()
{
 echo "${USERNAME}::1000:${USERNAME}"    >> /etc/group
 echo "${USERNAME}:*:1000:1000:${HOSTNAME} User:/home/${USERNAME}:/bin/bash" >> /etc/pass
 echo "${USERNAME}:*:11312:0:99999:7:::" >> /etc/shadow
}

# ---------------------------------------------------------------------------- #
# Usage: mountit src dst "options"

mountit()
{
 BUILTIN_FS="iso9660 ext2 vfat"
 if [ "${DEBUG}" = "yes" ]
 then
  for fs in ${BUILTIN_FS}; do
   test -b $1 && mount -t $fs $3 $1 $2 && return 0
  done
 else
  for fs in ${BUILTIN_FS}; do
   test -b $1 && mount -t $fs $3 $1 $2 >/dev/null 2>&1 && return 0
  done
 fi
 return 1
}

# ---------------------------------------------------------------------------- #
# Mount module disk

mountmodules()
{
 [ "${VERBOSE}" = "yes" ] && echo -n "${FMESSAGE1}"
 read a
 [ "${VERBOSE}" = "yes" ] && echo -n "${FMESSAGE2}"

 # Mount over /modules/others/
```

```
 if mountit /dev/fd0 /modules/others "-o ro"; then
  [ "${VERBOSE}" = "yes" ] && echo "${GREEN}OK.${NORMAL}"
  return 0
 fi
 [ "${VERBOSE}" = "yes" ] && echo "${FMESSAGE3}"
 return 1
}

# ---------------------------------------------------------------------------- #
# Unmount module disk

umountmodules()
{
 [ "${VERBOSE}" = "yes" ] && echo -n "${FMESSAGE4}"
 if [ "${DEBUG}" = "yes" ]
 then
  umount /modules/others
 else
  umount /modules/others 2>/dev/null
 fi
 [ "${VERBOSE}" = "yes" ] && echo "${DONEGREEN}"
}

# ---------------------------------------------------------------------------- #
# Ask user for modules

askmodules()
{
 MODULES="" # Reset variable
 TYPE="$1"; shift
 echo "${BLUE}${FMESSAGE5A} [ ${TYPE} ] ${FMESSAGE5B}"
 c=""; for m in "$@"; do
  if test -f "/modules/${TYPE}/$m"; then
   test -z "$c"  && { echo -n "    $m"; c="1"; } || { echo "               $m"; c=""; }
  fi
 done
 test -n "$c"  && echo ""
 echo "${FMESSAGE6A} [ ${TYPE} ] ${FMESSAGE6B}"
 echo "${FMESSAGE6C}"
 echo -n "${FMESSAGE7}"
 read MODULES
 case "${MODULES}" in
  n|N)
   MODULES="";
   ;;
  y|"")
   MODULES="$*";
   ;;
 esac
}

# ---------------------------------------------------------------------------- #
# Try to load the given modules (full path or current directory)
```

```
loadmodules()
{
 TYPE="$1"; shift
 [ "${VERBOSE}"     = "yes" ] && echo "0" > /proc/sys/kernel/printk
 [ "${DEBUG}"       = "yes" ] && echo "6" > /proc/sys/kernel/printk
 for i in "$@"; do
  [ "${VERBOSE}" = "yes" ] && echo -n "${FMESSAGE8} ${TYPE}... ${MAGENTA}${i}${NORMAL}"
  if test -f /modules/${TYPE}/$i && insmod -f /modules/${TYPE}/$i >/dev/null 2>&1
  then
   [ "${VERBOSE}" = "yes" ] && echo "${FMESSAGE9A} ${TYPE} ${FMESSAGE9B} \
    ${MAGENTA}${i}${GREEN}.${NORMAL}"
   case "$TYPE" in
    scsi|SCSI)
     FOUND_SCSI="yes";
     ;;
   esac
  fi
 done
 [ "${VERBOSE}"        = "yes" ] && echo -n "${CRE}"
 if [ "${FOUND_SCSI}"  = "yes" ]; then
  return 0
 else
  return 1
 fi
}

# ---------------------------------------------------------------------------- #
# Find /META directory in $DEVICES (CDROMs, HDs, etc.)

checkformetadirectory()
{
 FOUND_SCSI="$1"; shift

 # Check for ide-scsi supported CD-Roms et al.
 test -f /proc/scsi/scsi && FOUND_SCSI="yes"

 # Control kernel messages
 if [ "${DEBUG}" = "yes" ]
 then
  echo "6" > /proc/sys/kernel/printk
 else
  echo "0" > /proc/sys/kernel/printk
 fi

 # Now that the right SCSI driver is (hopefully) loaded, try to find CDROM
 DEVICES="/dev/hd?"
 [ "${FOUND_SCSI}" = "yes" ] && DEVICES="/dev/scd? /dev/scd?? ${DEVICES}"
 # New: Also try parallel port CD-Roms [for Mike].
 DEVICES="${DEVICES} /dev/pcd?"

 test -n "${FOUND_SCSI}" && DEVICES="${DEVICES} /dev/sd?[1-9] /dev/sd?[1-9][0-9]"
 DEVICES="${DEVICES} /dev/hd?[1-9] /dev/hd?[1-9][0-9]"
```

```
 for i in ${DEVICES}
 do
  [ "${VERBOSE}" = "yes" ] && echo -n "${FMESSAGE10} ${MAGENTA}${i}${NORMAL}   "
  if mountit $i /cdrom "-o ro";
  then
   if test -d /cdrom/META
   then
    # Found META directory in $i DEVICE, break loop and return $i
    [ "${VERBOSE}" = "yes" ] && echo -n "${FMESSAGE11} \
     ${MAGENTA}${i}${GREEN}...${NORMAL}"
    # Found META
    return 0
    break
   fi
   if [ "${DEBUG}" = "yes" ]
   then
    umount /cdrom
   else
    umount /cdrom >/dev/null 2>&1
   fi
  fi
 done

 # Not found META
 return 1
}

# ---------------------------------------------------------------------------- #
# Check for sufficient memory to mount extra ramdisk for /home + /var

checkformemory()
{

 # New in Kernel 2.4.x: tempfs with variable ramdisk size.
 # We check for available memory anyways and limit the ramdisks
 # to a reasonable size.
 #
 # Busybox don't have awk...
 #
 # FOUNDMEM="`awk '/MemTotal/{print $2}' /proc/meminfo`"
 # TOTALMEM="`awk 'BEGIN{m=0};/MemFree|Cached/{m+=$2};END{print m}' /proc/meminfo`"

 count="0"

 # Find available RAM memory
 for mem in `cat /proc/meminfo |grep "MemTotal:"`
 do
  count="`expr ${count} \+ 1`"
  [ "${count}" = "2" ] && FOUNDMEM="${mem}"
 done

 count="0" # Reset count
```

```
# Find Free RAM memory
for mem in ‘cat /proc/meminfo |grep "Cached:"‘
do
 count="‘expr ${count} \+ 1‘"
 [ "${count}" = "2" ] && cached="${mem}"
done

count="0" # Reset count

for mem in ‘cat /proc/meminfo |grep "MemFree:"‘
do
 count="‘expr ${count} \+ 1‘"
 [ "${count}" = "2" ] && memfree="${mem}"
done

TOTALMEN="‘expr ${cached} \+ ${memfree}‘"

# Be verbose if defined
[ "${VERBOSE}" = "yes" ] && echo "${FMESSAGE12} ${YELLOW}${FOUNDMEM}${BLUE} ${NORMAL}"

# Now we need to use a little intuition for finding a ramdisk size
# that keeps us from running out of space, but still doesn’t crash the
# machine due to lack of Ram

# Minimum size of additional ram partitions
MINSIZE="2000"
# At least this much memory minus 30% should remain when home and var are full.
MINLEFT="16000"
# Maximum ramdisk size
MAXSIZE="‘expr ${TOTALMEN} \- ${MINLEFT}‘"
# Default ramdisk size for ramdisk
RAMSIZE="‘expr ${TOTALMEN} \/ 5‘"

if test -n "${TOTALMEN}" -a "${TOTALMEN}" -gt "${MINLEFT}"
then
 test -z "${RAMSIZE}" && RAMSIZE=1000000
 mkdir -p /ramdisk
 # tmpfs/varsize version, can use swap
 RAMSIZE="‘expr $RAMSIZE \* 4‘"
 [ "${VERBOSE}" = "yes" ] && echo -n "${FMESSAGE13A}${RAMSIZE}${FMESSAGE13B}"
 # We need /bin/mount here for the -o size= option (busybox don’t have...this option :-(
 # /static/mount -t tmpfs -o "size=${RAMSIZE}k" /dev/shm /ramdisk && mkdir -p \
 # /ramdisk/home /ramdisk/var /ramdisk/dev /ramdisk/mnt && ln -s /ramdisk/home /ramdisk/
 if [ "${DEBUG}" = "yes" ]
 then
  /static/mount -t tmpfs /dev/shm /ramdisk && mkdir -p /ramdisk/home \
  /ramdisk/var /ramdisk/dev /ramdisk/mnt && ln -s /ramdisk/home /ramdisk/var /
 else
  /static/mount -t tmpfs /dev/shm /ramdisk >/dev/null 2>&1 && mkdir -p \
  /ramdisk/home /ramdisk/var /ramdisk/dev /ramdisk/mnt && ln -s \
  /ramdisk/home /ramdisk/var /
 fi
```

```
  if [ ! -e /ramdisk/dev/.devfsd ]
  then
   #mount devfs
   if [ "${DEBUG}" = "yes" ]
   then
    mount -t devfs devfs /ramdisk/dev
   else
    mount -t devfs devfs /ramdisk/dev >/dev/null 2>&1
   fi
  fi

  [ "${VERBOSE}" = "yes" ] && echo "${DONEBLUE}"
 else
  mkdir -p /home /var
 fi

}

# ---------------------------------------------------------------------------- #
# Mount (c)loop image

mountimage()
{
 if test "${FOUND_META}" != "no" -a -f /cdrom/META/${DISTRO}.cloop; then
  # Found compressed image (cloop)

  if [ "${DEBUG}" = "yes" ]
  then
   mknod /dev/cloop b 240 0
  else
   mknod /dev/cloop b 240 0 >/dev/null 2>/dev/null
  fi
  if [ "$CDCACHE" = "yes" ]
  then
   [ "${VERBOSE}" = "yes" ] && echo -e "${FMESSAGE14}"
   cp /cdrom/META/${DISTRO}.cloop /ramdisk/mnt/
   if [ $? -ne 0 ]
   then
    [ "${VERBOSE}" = "yes" ] && echo -e "${FMESSAGE15}"
    CDCACHE="no"
    rm -f /ramdisk/mnt/${DISTRO}.cloop
   else
    if [ "${DEBUG}" = "yes" ]
    then
     insmod -f /modules/cloop.o file=/cdrom/META/${DISTRO}.cloop
    else
     insmod -f /modules/cloop.o file=/cdrom/META/${DISTRO}.cloop \
     >/dev/null 2>&1
    fi
    mountit /dev/cloop /META "-o ro" || FOUND_META="no"
   fi
  fi
  if [ "$CDCACHE" = "no" ]
```

```
  then
   [ "${VERBOSE}" = "yes" ] && echo -e "${FMESSAGE16}"

   if [ "${DEBUG}" = "yes" ]
   then
    insmod -f /modules/cloop.o file=/cdrom/META/${DISTRO}.cloop
   else
    insmod -f /modules/cloop.o file=/cdrom/META/${DISTRO}.cloop >/dev/null 2>&1
   fi
   mountit /dev/cloop /META "-o ro" || FOUND_META="no"
  fi
 fi

 if test "${FOUND_META}" != "no" -a -f /cdrom/META/${DISTRO}.loop; then
  # Found loop image

  if [ "${DEBUG}" = "yes" ]
  then
   mknod /dev/loop0 b 7 0
  else
   mknod /dev/loop0 b 7 0 >/dev/null 2>/dev/null
  fi
  [ "${VERBOSE}" = "yes" ] && echo -e "${FMESSAGE17}"
  losetup /dev/loop0 /cdrom/META/${DISTRO}.loop
  mountit /dev/loop0 /META "-o ro" || FOUND_META="no"
 fi
}

# ------------------------------------------------------------------------- #
# Create common WRITABLE (empty) dirs

createdirs()
{
 [ "${VERBOSE}" = "yes" ] && echo -n "${FMESSAGE18}"

 # A metadistro don't will have all the files/directories list below

 # Show debug info?
 if [ "${DEBUG}" = "yes" ]
 then
  # Create common WRITABLE (empty) dirs
  /bin/mkdir -p /var/run /var/backups /var/cache/apache /var/local /var/lock/news \
   /var/nis /var/preserve /var/state/misc /var/tmp /var/lib \
   /var/spool/cups/tmp \
   /mnt/cdrom /mnt/floppy /mnt/hd /mnt/test \
   /home/${USERNAME} /root /etc/metadistros /etc/X11 /etc/cups
  /bin/chown ${USERNAME}.${USERNAME} /home/${USERNAME}
  # Create empty utmp and wtmp
  :> /var/run/utmp
  :> /var/run/wtmp
  # CUPS wants writable files. :-/
   /bin/cp -a /META/etc/cups/*.conf /etc/cups/
  # All files in here should be size zero after Knoppix.clean was run
```

```
 /bin/cp -a /META/var/local /META/var/games /META/var/log \
  /META/var/spool /var/
 /bin/cp -a /META/var/lib/games /META/var/lib/wine \
  /META/var/lib/nfs /META/var/lib/xkb /META/var/lib/isdn \
  /META/var/lib/kdm /META/var/lib/pcmcia \
  /META/var/lib/dhcp* \
  /var/lib/
 # Problematic directories in /var/lib (lots and lots of inodes)
 /bin/ln -s /META/var/lib/dpkg /META/var/lib/apt /META/var/lib/doc-base \
  /META/var/lib/gnome /META/var/lib/kde \
  /META/var/lib/scrollkeeper /META/var/lib/texmf \
  /var/lib/
 # Debian-apt
 /bin/ln -s /META/var/cache/apt /var/cache/
 /bin/ln -s /META/etc/skel /META/etc/nessus /etc/dhcpc/resolv.conf \
  /etc/
 /bin/ln -s /META/dev/* /dev/
 # Index files can be HUGE, so better replace cache/man tree by links later
 # cp -a /META/var/cache/man /var/cache/ [ "${DEBUG}" = "no" ] && 2>/dev/null
 # Create links from CDROM for UNWRITABLE (remaining) files
 /bin/cp -aus /META/var/* /var/
 /bin/cp -aus /META/etc/* /etc/
 # Make SURE that these are files, not links!
 /bin/rm -rf /etc/ftpusers /etc/passwd /etc/shadow /etc/group \
  /etc/ppp /etc/isdn /etc/ssh /etc/ioctl.save \
  /etc/inittab /etc/network /etc/sudoers \
  /etc/init /etc/localtime /etc/dhcpc /etc/pnm2ppa.conf
 /bin/cp -a /etc/ftpusers /META/etc/passwd /META/etc/shadow /META/etc/group \
  /META/etc/ppp /META/etc/isdn /META/etc/ssh \
  /META/etc/inittab /META/etc/network /META/etc/sudoers \
  /META/sbin/init /META/etc/dhcpc /etc/
 # Extremely important, init crashes on shutdown if this is only a link
 :> /etc/ioctl.save
 :> /etc/pnm2ppa.conf
 # Diet libc bug workaround
 /bin/cp -f /META/etc/localtime /etc/localtime

else
 # Create common WRITABLE (empty) dirs
 /bin/mkdir -p /var/run /var/backups /var/cache/apache /var/local /var/lock/news \
  /var/nis /var/preserve /var/state/misc /var/tmp /var/lib \
  /var/spool/cups/tmp \
  /mnt/cdrom /mnt/floppy /mnt/hd /mnt/test \
  /home/${USERNAME} /root /etc/metadistros /etc/X11 /etc/cups >/dev/null 2>&1
 /bin/chown ${USERNAME}.${USERNAME} /home/${USERNAME} >/dev/null 2>&1
 # Create empty utmp and wtmp
 :> /var/run/utmp >/dev/null 2>&1
 :> /var/run/wtmp >/dev/null 2>&1
 # CUPS wants writable files. :-/
 /bin/cp -a /META/etc/cups/*.conf /etc/cups/ >/dev/null 2>&1
 # All files in here should be size zero after Knoppix.clean was run
 /bin/cp -a /META/var/local /META/var/games /META/var/log \
  /META/var/spool /var/ >/dev/null 2>&1
```

```
  /bin/cp -a /META/var/lib/games /META/var/lib/wine \
   /META/var/lib/nfs /META/var/lib/xkb /META/var/lib/isdn \
   /META/var/lib/kdm /META/var/lib/pcmcia \
   /META/var/lib/dhcp* \
   /var/lib/ >/dev/null 2>&1
  # Problematic directories in /var/lib (lots and lots of inodes)
  /bin/ln -s /META/var/lib/dpkg /META/var/lib/apt /META/var/lib/doc-base \
   /META/var/lib/gnome /META/var/lib/kde \
   /META/var/lib/scrollkeeper /META/var/lib/texmf \
   /var/lib/ >/dev/null 2>&1
  # Debian-apt
  /bin/ln -s /META/var/cache/apt /var/cache/ >/dev/null 2>&1
  /bin/ln -s /META/etc/skel /META/etc/nessus /etc/dhcpc/resolv.conf \
   /etc/ >/dev/null 2>&1
  /bin/ln -s /META/dev/* /dev/ >/dev/null 2>&1
  # Index files can be HUGE, so better replace cache/man tree by links later
  # cp -a /META/var/cache/man /var/cache/ [ "${DEBUG}" = "no" ] && 2>/dev/null
  # Create links from CDROM for UNWRITABLE (remaining) files
  /bin/cp -aus /META/var/* /var/ >/dev/null 2>&1
  /bin/cp -aus /META/etc/* /etc/ >/dev/null 2>&1
  # Make SURE that these are files, not links!
  /bin/rm -rf /etc/ftpusers /etc/passwd /etc/shadow /etc/group \
   /etc/ppp /etc/isdn /etc/ssh /etc/ioctl.save \
   /etc/inittab /etc/network /etc/sudoers \
   /etc/init /etc/localtime /etc/dhcpc /etc/pnm2ppa.conf >/dev/null 2>&1
  /bin/cp -a /etc/ftpusers /META/etc/passwd /META/etc/shadow /META/etc/group \
   /META/etc/ppp /META/etc/isdn /META/etc/ssh \
   /META/etc/inittab /META/etc/network /META/etc/sudoers \
   /META/sbin/init /META/etc/dhcpc /etc/ >/dev/null 2>&1
  # Extremely important, init crashes on shutdown if this is only a link
  :> /etc/ioctl.save >/dev/null 2>&1
  :> /etc/pnm2ppa.conf >/dev/null 2>&1
  # Diet libc bug workaround
  /bin/cp -f /META/etc/localtime /etc/localtime >/dev/null 2>&1
 fi

 [ "${VERBOSE}" = "yes" ] && echo "${DONEBLUE}"
}


# ---------------------------------------------------------------------------- #
# Set keymap: Get and set keymap configuration from 'var.conf'

setkeymap() {
 if [ -n ${KEYMAP} ]
        then
                [ "${VERBOSE}" = "yes" ] && echo "${BLUE}Loading ${KEYMAP} keymap${NORMAL
                loadkmap < /keymaps/${KEYMAP}.map
        elif
        then
                #default keymap is "us"
                [ "${VERBOSE}" = "yes" ] && echo "${BLUE}Loading default (US) keymap${NOR
                loadkmap < /keymaps/us.map
```

```
        fi
}


# ------------------------------------------------------------------------- #
# Control progress bar
#
# Usage: progressbar number text   <-- number = %   text = message display
#        progressbar f             <-- fail
#        progressbar w             <-- warning
#

progressbar ()
{
 # Accept 1 or 2 parameters
 if [ "$#" -gt "0" ] && [ "$#" -lt "3" ] && [ "${VERBOSE}" = "no" ] && [ -f /proc/progres
 then
  # Normal message
  [ "$#" -eq "2" ] && ( echo ${1} ${2} > /proc/progress; return 0; )

  # Warning or fail message
  ( [ ${1} = 'w' ]  || [ ${1} = 'f' ] ) && ( echo ${1} > /proc/progress; return 0; )
 else
  return 1 # fail
        fi
}
```

C. i18n.conf

```
#!/static/sh
#
# Metadistros boot up script - internationalitation file
#
# Copyright 2003 Metadistros [ http://metadistros.hispalinux.es/ ]
# Distributed under the GPL

# Load configuration file
. /etc/metadistros/var.conf

# Messages don't set by default... (If make a translation set
# this variable to "yes" in your language settings)
SET_MESSAGES="no"

##
# The default language/keyboard to use. This CANNOT be autoprobed.
# Most of these variables will be used to generate the KDE defaults
#
# MESSAGE*  = messages in linuxrc file
# FMESSAGE* = messages in functions file
# PBMESSAGE* = progress bar messages
```

```
#

case "$LANGUAGE" in
 es)
  # Spanish version
  COUNTRY="es"
  LANG="es_ES@euro"
  KEYTABLE="es"
  KEYMAP="es"
  XKEYBOARD="es"
  KDEKEYBOARD="es"
  CHARSET="iso8859-15"
  # Additional KDE Keyboards
  KDEKEYBOARDS="de,us,fr"

  # Spanish messages
  WELCOME="¡Bienvenido a ${DISTRO}!"
  DONEBLUE="${BLUE}Hecho.${NORMAL}"
  DONEGREEN="${GREEN}Hecho.${NORMAL}"
  MESSAGE1="${CYAN}¿Quiere cargar algún módulo adicional desde un/otro disquete? \
   [${WHITE}Y${CYAN}/n] ${NORMAL}"
  MESSAGE2="${CRE}${BLUE}Arrancando el proceso init.${NORMAL}"
  MESSAGE3="${CRE}${RED}Lo siento, no puedo encontrar el sistema de ficheros META.\n \
   Le dejo ante un intérprete de comandos (muy limitado).\n\n - Teclee 'help' \
   para ver una lista de los comandos disponibles\n - Pulse 'reset' para \
   salir.${NORMAL}\n"
  FMESSAGE1="${CRE}${CYAN}Por favor, inserte el disco de módulos y pulse Intro. ${NORMAL}
  FMESSAGE2="${CRE}${BLUE}Montando el disco de módulos... ${NORMAL}"
  FMESSAGE3="${RED}NO ENCONTRADO.${NORMAL}"
  FMESSAGE4="${CRE}${BLUE}Desmontando el disco de módulos... ${NORMAL}"
  FMESSAGE5A="${BLUE}módulos"
  FMESSAGE5B="disponibles:${WHITE}"
  FMESSAGE6A="${CYAN}¿Cargar módulos"
  FMESSAGE6B="?${NORMAL}"
  FMESSAGE6C="${CYAN}[Teclee el nombre completo del fichero(s) (separado por espacios), \
   pulse Intro para auto-prueba, ${WHITE}n${CYAN} para nada] ${NORMAL}"
  FMESSAGE7="${CYAN}módulo(s) a instalar> ${NORMAL}"
  FMESSAGE8="${CRE}${BLUE}Probando"
  FMESSAGE9A="${CRE} ${GREEN}Encontrado dispositivo(s)"
  FMESSAGE9B="controlado(s) por"
  FMESSAGE10="${CRE}${BLUE}Buscando CDROM en:${NORMAL}"
  FMESSAGE11="${CRE}${GREEN}Accediendo al CDROM '${DISTRO}' en${NORMAL}"
  FMESSAGE12="${CRE}${BLUE}Memoria total encontrada (en kB):${NORMAL}"
  FMESSAGE13A="${CRE}${BLUE}Creando ${YELLOW}/ramdisk${BLUE} (tamaño dinámico="
  FMESSAGE13B="k) en ${MAGENTA}/dev/shm${BLUE}...${NORMAL}"
  FMESSAGE14="${BLUE}Copiando la imagen del CD a tmpfs${NORMAL}"
  FMESSAGE15="${RED}La copia ha fallado (posiblemente por falta de espacio)${NORMAL}"
  FMESSAGE16="${BLUE}Montando el loopback comprimido${NORMAL}"
  FMESSAGE17="${BLUE}Montando el loopback${NORMAL}"
  FMESSAGE18="${CRE}${BLUE}Creando directorios y enlaces simbólicos en ramdisk...${NORMA
  # Progress Bar Messages
  PBMESSAGE1="Cargando módulos FIREWIRE"
  PBMESSAGE2="Cargando módulos ATARAID"
```

```
 PBMESSAGE3="Cargando módulos PCMCIA"
 PBMESSAGE4="Cargando módulos USB"
 PBMESSAGE5="Cargando módulos SCSI"
 PBMESSAGE6="Buscando el directorio META"
 PBMESSAGE7="Buscando MEMORIA"
 PBMESSAGE8="Montando la imagen (c)loop"
 PBMESSAGE9="INIT tomará el control en unos segundos"

 # Messages are set...
 SET_MESSAGES="yes"
 ;;
de)
 # German version
 LANG="de_DE@euro"
 LANGUAGE="de_DE@euro"
 COUNTRY="de"
 KEYTABLE="de-latin1-nodeadkeys"
 KEYMAP="de"
 XKEYBOARD="de"
 KDEKEYBOARD="de"
 CHARSET="iso8859-15"
 # Additional KDE Keyboards
 KDEKEYBOARDS="us,fr"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
be)
 # BE version
 LANG="be"
 LANGUAGE="be"
 COUNTRY="be"
 KEYTABLE="be-latin1"
 KEYMAP="be"
 XKEYBOARD="be"
 KDEKEYBOARD="be"
 CHARSET="iso8859-15"
 # Additional KDE Keyboards
 KDEKEYBOARDS="us,de,fr"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
bg)
 # BG version
 LANG="bg_BG"
 LANGUAGE="bg"
 COUNTRY="bg"
 KEYTABLE="bg"
 KEYMAP="bg"
 XKEYBOARD="bg"
 KDEKEYBOARD="bg"
 CHARSET="microsoft-cp1251"
```

```
 # Additional KDE Keyboards
 KDEKEYBOARDS="us,de,fr"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
ch)
 # Swiss version (basically de with some modifications)
 LANG="de_CH"
 LANGUAGE="de"
 COUNTRY="ch"
 KEYTABLE="sg-latin1"
 KEYMAP="de"
 XKEYBOARD="de_CH"
 KDEKEYBOARD="de_CH"
 CHARSET="iso8859-15"
 # Additional KDE Keyboards
 KDEKEYBOARDS="de,us,fr"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
cn)
 # Simplified Chinese version
 COUNTRY="cn"
 LANG="zh_CN.GB2312"
 KEYTABLE="us"
 KEYMAP="us"
 XKEYBOARD="us"
 KDEKEYBOARD="us"
 CHARSET="gb2312.1980-0"
 # Additional KDE Keyboards
 KDEKEYBOARDS="us,de,fr"
 XMODIFIERS="@im=Chinput"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
cs|cz)
 # Czech version
 LANGUAGE="cs"
 COUNTRY="cs"
 LANG="cs_CZ"
 KEYTABLE="cz-lat2"
 KEYMAP="cs"
 XKEYBOARD="cs"
 KDEKEYBOARD="cs"
 CHARSET="iso8859-2"
 # Additional KDE Keyboards
 KDEKEYBOARDS="us,de,fr"

 # There are a translation for this language?
 SET_MESSAGES="no"
```

```
 ;;
dk|da)
 # Dansk version
 COUNTRY="dk"
 LANG="da_DK"
 # Workaround: "dk" broken in gettext, use da:da_DK
 LANGUAGE="da:da_DK"
 # Keytable "dk" is correct.
 KEYTABLE="dk"
 KEYMAP="dk"
 XKEYBOARD="dk"
 KDEKEYBOARD="dk"
 CHARSET="iso8859-15"
 # Additional KDE Keyboards
 KDEKEYBOARDS="dk,de,us,fr"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
fi)
 # finnish version, though we may not have the kde-i18n files
 COUNTRY="fi"
 LANG="fi_FI@euro"
 KEYTABLE="fi"
 KEYMAP="fi"
 XKEYBOARD="fi"
 KDEKEYBOARD="fi"
 CHARSET="iso8859-15"
 # Additional KDE Keyboards
 KDEKEYBOARDS="us"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
fr)
 # french version
 COUNTRY="fr"
 LANG="fr_FR@euro"
 KEYTABLE="fr"
 KEYMAP="fr"
 XKEYBOARD="fr"
 KDEKEYBOARD="fr"
 CHARSET="iso8859-15"
 # Additional KDE Keyboards
 KDEKEYBOARDS="de,us"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
it)
 # italian version
 COUNTRY="it"
 LANG="it_IT@euro"
```

```
 KEYTABLE="it"
 KEYMAP="it"
 XKEYBOARD="it"
 KDEKEYBOARD="it"
 CHARSET="iso8859-15"
 # Additional KDE Keyboards
 KDEKEYBOARDS="fr,us,de"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
ja)
 # (limited) japanese version
 COUNTRY="jp"
 LANG="ja_JP"
 LANGUAGE="ja"
 KEYTABLE="us"
 KEYMAP="jp"
 XKEYBOARD="us"
 KDEKEYBOARD="us"
 CHARSET="iso8859-15"
 # Additional KDE Keyboards
 KDEKEYBOARDS="fr,us,de"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
nl)
 # netherland version
 COUNTRY="nl"
 LANG="nl_NL@euro"
 KEYTABLE="us"
 KEYMAP="nl"
 XKEYBOARD="us"
 KDEKEYBOARD="en_US"
 CHARSET="iso8859-15"
 # Additional KDE Keyboards
 KDEKEYBOARDS="nl,de,fr"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
pl)
 # Polish version
 COUNTRY="pl"
 LANG="pl_PL"
 KEYTABLE="pl"
 KEYMAP="pl"
 XKEYBOARD="pl"
 KDEKEYBOARD="pl"
 CHARSET="iso8859-2"
 # Additional KDE Keyboards
 KDEKEYBOARDS="de,us,fr"
```

```
 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
ru)
 # Russian version
 COUNTRY="ru"
 LANG="ru_RU.KOI8-R"
 KEYTABLE="ru"
 KEYMAP="ru"
 XKEYBOARD="ru"
 KDEKEYBOARD="ru"
 CHARSET="koi8-r"
 CONSOLEFONT="Cyr_a8x16"
 # Additional KDE Keyboards
 KDEKEYBOARDS="de,us,fr"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
sk)
 # Slovak version (guessed)
 COUNTRY="sk"
 LANG="sk"
 KEYTABLE="sk-qwerty"
 KEYMAP="sk-y"
 XKEYBOARD="sk"
 KDEKEYBOARD="sk"
 CHARSET="iso8859-2"
 # Additional KDE Keyboards
 KDEKEYBOARDS="us,de,fr"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
tr)
 # Turkish version (guessed)
 COUNTRY="tr"
 LANG="tr_TR"
 KEYTABLE="tr_q-latin5"
 KEYMAP="trq"
 XKEYBOARD="tr"
 KDEKEYBOARD="tr"
 CHARSET="iso8859-9"
 # Additional KDE Keyboards
 KDEKEYBOARDS="us,de,fr"

 # There are a translation for this language?
 SET_MESSAGES="no"
 ;;
tw)
 # Traditional chinese version (thanks to Chung-Yen Chang)
 COUNTRY="tw"
```

```
  LANG="zh_TW.Big5"
  LANGUAGE="zh_TW.Big5"
  KEYTABLE="us"
  KEYMAP="us"
  XKEYBOARD="us"
  KDEKEYBOARD="us"
  # CHARSET="big5-0"
  CHARSET="iso8859-1"
  # Additional KDE Keyboards
  KDEKEYBOARDS="us"
  XMODIFIERS="@im=xcin"

  # There are a translation for this language?
  SET_MESSAGES="no"
  ;;
 uk)
  # british version
  COUNTRY="uk"
  LANG="en_GB"
  KEYTABLE="uk"
  KEYMAP="uk"
  XKEYBOARD="uk"
  KDEKEYBOARD="uk"
  CHARSET="iso8859-1"
  # Additional KDE Keyboards
  KDEKEYBOARDS="us"

  # There are a translation for this language?
  SET_MESSAGES="no"
  ;;
 *)
  # US version
  LANGUAGE="us"
  COUNTRY="us"
  LANG="C"
  KEYTABLE="us"
  KEYMAP="us"
  XKEYBOARD="us"
  KDEKEYBOARD="us"
  CHARSET="iso8859-1"
  # Additional KDE Keyboards
  KDEKEYBOARDS="de,fr"

  # There are a translation for this language?
  SET_MESSAGES="no"
  ;;
esac


##
# Set default messages, if don't set...
#
```

```
if [ ${SET_MESSAGES} = "no" ]
then
  # Default messages
  WELCOME="Welcome to ${DISTRO}!"
  DONEBLUE="${BLUE}Done.${NORMAL}"
  DONEGREEN="${GREEN}Done.${NORMAL}"
  MESSAGE1="${CYAN}Do you want to load additional modules from (another) floppy disk? \
   [${WHITE}Y${CYAN}/n] ${NORMAL}"
  MESSAGE2="${CRE}${BLUE}Starting init process.${NORMAL}"
  MESSAGE3="${CRE}${RED}Can't find META filesystem, sorry.\nDropping you to a (very \
   limited) shell.\n\n - Enter 'help' for a list of built-in commands\n - \
   Press 'reset' button to quit.${NORMAL}\n"
  FMESSAGE1="${CRE}${CYAN}Please insert modules disk and hit Return. ${NORMAL}"
  FMESSAGE2="${CRE}${BLUE}Mounting modules disk... ${NORMAL}"
  FMESSAGE3="${RED}NOT FOUND.${NORMAL}"
  FMESSAGE4="${CRE}${BLUE}Unmounting modules disk... ${NORMAL}"
  FMESSAGE5A="" # Not necesary in English
  FMESSAGE5B="${BLUE}modules available:${WHITE}"
  FMESSAGE6A="${CYAN}Load"
  FMESSAGE6B="Modules?${NORMAL}"
  FMESSAGE6C="${CYAN}[Enter full filename(s) (space-separated), Return for autoprobe, \
   ${WHITE}n${CYAN} for none] ${NORMAL}"
  FMESSAGE7="${CYAN}insmod module(s)> ${NORMAL}"
  FMESSAGE8="${CRE}${BLUE}Probing"
  FMESSAGE9A="${CRE} ${GREEN}Found"
  FMESSAGE9B="device(s) handled by"
  FMESSAGE10="${CRE}${BLUE}Looking for CDROM in:${NORMAL}"
  FMESSAGE11="${CRE} ${GREEN}Accessing ${DISTRO} CDROM at${NORMAL}"
  FMESSAGE12="${CRE}${BLUE}Total memory found (in kB):${NORMAL}"
  FMESSAGE13A="${CRE}${BLUE}Creating ${YELLOW}/ramdisk${BLUE} (dynamic size="
  FMESSAGE13B="k) on ${MAGENTA}/dev/shm${BLUE}...${NORMAL}"
  FMESSAGE14="${BLUE}Attempting to cache CD image to tmpfs${NORMAL}"
  FMESSAGE15="${RED}Caching failed (likely due to lack of space)${NORMAL}"
  FMESSAGE16="${BLUE}Mounting compressed loopback filesystem${NORMAL}"
  FMESSAG17="${BLUE}Mounting loopback filesystem${NORMAL}"
  FMESSAGE18="${CRE}${BLUE}Creating directories and symlinks on ramdisk...${NORMAL}"
  # Progress Bar Messages
  PBMESSAGE1="Load FIREWIRE modules"
  PBMESSAGE2="Load ATARAID modules"
  PBMESSAGE3="Load PCMCIA modules"
  PBMESSAGE4="Load USB modules"
  PBMESSAGE5="Load SCSI modules"
  PBMESSAGE6="Check for META directory"
  PBMESSAGE7="Check for MEMORY"
  PBMESSAGE8="Mount (c)loop image"
  PBMESSAGE9="INIT will take control in few seconds"
fi
```

D. var.conf

```
#!/static/sh
#
# Metadistros boot up script - configuration file
#
# Copyright 2003 Metadistros [ http://metadistros.hispalinux.es/ ]
# Distributed under the GPL


##
# Read boot command line
#
CMDLINE="‘cat /proc/cmdline‘"

##
# This variables can be set in comand line
#

 ##
 # variable=value
 #

  LANGUAGE="‘echo ${lang}‘"
  [ -n "${LANGUAGE}" ] || LANGUAGE="es" #<- Set language here

  RAMSIZE="‘echo ${ramsize}‘"
  [ -n "${RAMSIZE}" ] || RAMSIZE="1000000" # Default maximum size of dynamic
        # ramdisk in kilobytes

  KEYMAP="‘echo ${keymap}‘"
  [ -n "${KEYMAP}" ] || KEYMAP="es"

  ##
  # Network configuration
  #

   IP="‘echo ${ip}‘"
   [ -n "${IP}" ] || IP=""

   BROADCAST="‘echo ${broadcast}‘"
   [ -n "${BROADCAST}" ] || BROADCAST=""

   NETMASK="‘echo ${netmask}‘"
   [ -n "${NETMASK}" ] || NETMASK=""

   GATEWAY="‘echo ${gateway}‘"
   [ -n "${GATEWAY}" ] || GATEWAY=""

 ##
 # value
 #
```

```
##
# Default value
#

 ##
 # What information display?
 #
 VERBOSE="no" # Show information in boot up?
 DEBUG="no" # Show debug information in boot up?
 INTERACTIVE="no" # By default run in quiet mode

 ##
 # Run the installation program?
 #
 INSTALL="no"

 ##
 # Hardware for check
 #
 SCSI="yes" # Scan for scsi devices
 FIREWIRE="no" # Modprobes firewire modules in initrd (for firewire cdroms,etc)
 ATARAID="no" # Loads ide raid modules from initrd
 PCMCIA="no" # Starts pcmcia service
 USB="yes" # Disables usb module load from initrd, disables hotplug

 CDCACHE="no" # Cache the entire runtime portion of cd in RAM. This
       # uses a lot of RAM (depend on ), but allows you to umount /mnt/cdrom
       # and mount another cdrom.

##
# Check and set above variables
#

for x in ${CMDLINE}
do
 case "${x}" in
  *verbose*)
                VERBOSE="yes";
   exec >/dev/console </dev/console 2>&1; # Clean input/output
   echo "${CLEAR}"; # Clear and reset Screen
   ;;
  *showdebug*)
   DEBUG="yes";
                VERBOSE="yes";
   exec >/dev/console </dev/console 2>&1; # Clean input/output
   echo "${CLEAR}"; # Clear and reset Screen
   echo "6" > /proc/sys/kernel/printk # Enable kernel messages to console
   ;;
  *interactive*)
   INTERACTIVE="yes";
   VERBOSE="yes"; # Set verbose mode
   exec >/dev/console </dev/console 2>&1; # Clean input/output
   echo "${CLEAR}"; # Clear and reset Screen
```

```
     ;;
    *install*)
     INSTALL="yes";
     ;;
    *noscsi*)
     SCSI="no";
     ;;
    *dofirewire*)
     FIREWIRE="yes";
     ;;
    *nousb*)
     USB="no";
     ;;
    *doataraid*)
     ATARAID="yes";
     ;;
    *dopcmcia*)
     PCMCIA="yes";
     ;;
    *cdcache*)
     CDCACHE="yes";
     ;;
   esac
  done


##
# Misc variables
#

 # Name of metadistro
 DISTRO="SBM"
 # User for metadistro
 USERNAME="metadistros"
 # Hostname of metadistro
 HOSTNAME="metadistros"

 # Default value (don't touch)
 FOUND_SCSI="no"
 # Default value (don't touch)
 FOUND_META="no"

##
# 'Screen' variables
#

 # Reset fb color mode
 RESET="^[]R"
 # ANSI COLORS
 # Erase to end of line
 CRE="^M^[[K"
 # Clear and reset Screen
 CLEAR="^[c"
```

```
 # Normal color
 NORMAL="^[[0;39m"
 # RED: Failure or error message
 RED="^[[1;31m"
 # GREEN: Success message
 GREEN="^[[1;32m"
 # YELLOW: Descriptions
 YELLOW="^[[1;33m"
 # BLUE: System messages
 BLUE="^[[1;34m"
 # MAGENTA: Found devices or drivers
 MAGENTA="^[[1;35m"
 # CYAN: Questions
 CYAN="^[[1;36m"
 # BOLD WHITE: Hint
 WHITE="^[[1;37m"

##
# "Safe" SCSI modules in the right order for autoprobe
# Warning: The sym53c8xx.o and g_NCR* cause a kernel Oops if no such adapter
# is present.
#
# NB: It looks like that ncr53c8xx.o is more stable than 53c7,8xx.o for
# a ncr53c810 controller (at least on my installation box it's more
# immune to SCSI timeouts)
# Removed 53c7,8xx -> crashes if no device attached.
# Removed AM53C974 -> crashes tmscsim if adapter found
# Added initio.o on request (untested)
#
# (don't modify if you don't know what are you doing)
#
SCSI_MODULES="aic7xxx.o BusLogic.o \
ncr53c8xx.o NCR53c406a.o \
initio.o \
advansys.o aha1740.o aha1542.o aha152x.o \
atp870u.o dtc.o eata.o fdomain.o gdth.o \
megaraid.o pas16.o pci2220i.o pci2000.o psi240i.o \
qlogicfas.o qlogicfc.o qlogicisp.o \
seagate.o t128.o tmscsim.o u14-34f.o ultrastor.o wd7000.o \
a100u2w.o 3w-xxxx.o"

##
# Other modules
#

FIREWIRE_MODULES="ieee1394.o ohci1394.o eth1394.o sbp2.o"

ATARAID_MODULES="ataraid.o pdcraid.o hptraid.o"

PCMCIA_MODULES="pcmcia_core.o i82365.o ds.o ide-cs.o"

USB_MODULES="usbcore.o ehci-hcd.o uhci.o usb-ohci.o hid.o usb-storage.o"
```

E. Config.h-static

```
/* vi: set sw=4 ts=4: */
// This file defines the feature set to be compiled into busybox.
// When you turn things off here, they won't be compiled in at all.
//
//// This file is parsed by sed.  You MUST use single line comments.
//   i.e.,  //#define BB_BLAH
//
//
// BusyBox Applications
//#define BB_ADJTIMEX
#define BB_AR
#define BB_ASH
#define BB_BASENAME
#define BB_CAT
#define BB_CHGRP
#define BB_CHMOD
#define BB_CHOWN
#define BB_CHROOT
#define BB_CHVT
#define BB_CLEAR
#define BB_CMP
#define BB_CP
#define BB_CPIO
#define BB_CUT
#define BB_DATE
#define BB_DC
#define BB_DD
#define BB_DEALLOCVT
#define BB_DF
#define BB_DIRNAME
#define BB_DMESG
#define BB_DOS2UNIX
#define BB_DPKG
#define BB_DPKG_DEB
#define BB_DUTMP
#define BB_DU
#define BB_DUMPKMAP
#define BB_ECHO
#define BB_ENV
#define BB_EXPR
#define BB_FBSET
#define BB_FDFLUSH
#define BB_FIND
#define BB_FREE
#define BB_FREERAMDISK
#define BB_FSCK_MINIX
#define BB_GETOPT
#define BB_GREP
#define BB_GUNZIP
#define BB_GZIP
#define BB_HALT
```

```
#define BB_HEAD
#define BB_HOSTID
#define BB_HOSTNAME
//#define BB_HUSH
#define BB_ID
#define BB_IFCONFIG
#define BB_INIT
#define BB_INSMOD
#define BB_KILL
#define BB_KILLALL
#define BB_KLOGD
//#define BB_LASH
#define BB_LENGTH
#define BB_LN
#define BB_LOADACM
#define BB_LOADFONT
#define BB_LOADKMAP
#define BB_LOGGER
#define BB_LOGNAME
#define BB_LOSETUP
#define BB_LS
#define BB_LSMOD
#define BB_MAKEDEVS
#define BB_MD5SUM
#define BB_MKDIR
#define BB_MKFIFO
#define BB_MKFS_MINIX
#define BB_MKNOD
#define BB_MKSWAP
#define BB_MKTEMP
#define BB_MODPROBE
#define BB_MORE
#define BB_MOUNT
//#define BB_MSH
#define BB_MT
#define BB_MV
#define BB_NC
#define BB_NSLOOKUP
#define BB_PIDOF
#define BB_PING
#define BB_PIVOT_ROOT
#define BB_POWEROFF
#define BB_PRINTF
#define BB_PS
#define BB_PWD
#define BB_RDATE
#define BB_READLINK
#define BB_REBOOT
#define BB_RENICE
#define BB_RESET
#define BB_RM
#define BB_RMDIR
#define BB_RMMOD
```

```
#define BB_ROUTE
#define BB_RPM2CPIO
#define BB_SED
#define BB_SETKEYCODES
#define BB_SLEEP
#define BB_SORT
#define BB_STTY
#define BB_SWAPONOFF
#define BB_SYNC
#define BB_SYSLOGD
#define BB_TAIL
#define BB_TAR
#define BB_TEE
#define BB_TEST
#define BB_TELNET
#define BB_TFTP
#define BB_TIME
#define BB_TOP
#define BB_TOUCH
#define BB_TR
#define BB_TRACEROUTE
#define BB_TRUE_FALSE
#define BB_TTY
#define BB_UNIX2DOS
#define BB_UUENCODE
#define BB_UUDECODE
#define BB_UMOUNT
#define BB_UNIQ
#define BB_UNAME
#define BB_UPDATE
#define BB_UPTIME
#define BB_USLEEP
#define BB_VI
#define BB_WATCHDOG
#define BB_WC
#define BB_WGET
#define BB_WHICH
#define BB_WHOAMI
#define BB_XARGS
#define BB_YES
// End of Applications List
//
//
//
// ------------------------------------------------------
// This is where feature definitions go.  Generally speaking,
// turning this stuff off makes things a bit smaller (and less
// pretty/useful).
//
//
// If you enabled one or more of the shells, you may select which one
// should be run when sh is invoked:
#define BB_FEATURE_SH_IS_ASH
```

```
//#define BB_FEATURE_SH_IS_HUSH
//#define BB_FEATURE_SH_IS_LASH
//#define BB_FEATURE_SH_IS_MSH
//
// BusyBox will, by default, malloc space for its buffers.  This costs code
// size for the call to xmalloc.  You can use the following feature to have
// them put on the stack.  For some very small machines with limited stack
// space, this can be deadly.  For most folks, this works just fine...
//#define BB_FEATURE_BUFFERS_GO_ON_STACK
// The third alternative for buffer allocation is to use BSS.  This works
// beautifully for computers with a real MMU (and OS support), but wastes
// runtime RAM for uCLinux.  This behavior was the only one available for
// BusyBox versions 0.48 and earlier.
//#define BB_FEATURE_BUFFERS_GO_IN_BSS
//
// Turn this on to use Erik's very cool devps, and devmtab kernel drivers,
// thereby eliminating the need for the /proc filesystem and thereby saving
// lots and lots memory for more important things.  NOTE:  If you enable this
// feature, you _must_ have patched the kernel to include the devps patch that
// is included in the busybox/kernel-patches directory.  You will also need to
// create some device special files in /dev on your embedded system:
//        mknod /dev/mtab c 10 22
//        mknod /dev/ps c 10 21
// I emailed Linus and this patch will not be going into the stock kernel.
//#define BB_FEATURE_USE_DEVPS_PATCH
//
// show verbose usage messages
#define BB_FEATURE_VERBOSE_USAGE
//
// Use termios to manipulate the screen ('more' is prettier with this on)
#define BB_FEATURE_USE_TERMIOS
//
// calculate terminal & column widths (for more, ls, and telnet)
#define BB_FEATURE_AUTOWIDTH
//
// show username/groupnames for ls
#define BB_FEATURE_LS_USERNAME
//
// show file timestamps in ls
#define BB_FEATURE_LS_TIMESTAMPS
//
// enable ls -p and -F
#define BB_FEATURE_LS_FILETYPES
//
// sort the file names
#define BB_FEATURE_LS_SORTFILES
//
// enable ls -R
#define BB_FEATURE_LS_RECURSIVE
//
// enable ls -L
#define BB_FEATURE_LS_FOLLOWLINKS
//
```

```
// Use color to identify different file types
#define BB_FEATURE_LS_COLOR
//
// Disable for a smaller (but less functional) ping
#define BB_FEATURE_FANCY_PING
//
// Make init use a simplified /etc/inittab file (recommended).
#define BB_FEATURE_USE_INITTAB
//
//Enable init being called as /linuxrc
#define BB_FEATURE_LINUXRC
//
//Have init enable core dumping for child processes (for debugging only)
//#define BB_FEATURE_INIT_COREDUMPS
//
//Make sure nothing is printed to the console on boot
#define BB_FEATURE_EXTRA_QUIET
//
// enable syslogd -R remotehost
#define BB_FEATURE_REMOTE_LOG
//
// enable syslogd -C
//#define BB_FEATURE_IPC_SYSLOG
//
//Disable for a simple tail implementation (2.34k vs 3k for the full one).
//Both provide 'tail -f', but this cuts out -c, -q, -s, and -v.
#define BB_FEATURE_FANCY_TAIL
//
// Enable support for loop devices in mount
#define BB_FEATURE_MOUNT_LOOP
//
// Enable support for a real /etc/mtab file instead of /proc/mounts
//#define BB_FEATURE_MTAB_SUPPORT
//
// Enable support for mounting remote NFS volumes.
// You may need to mount with "-o nolock" if you are
// not running a local portmapper daemon...
//
// If you are using uClibc, be sure that you've already compiled
// uClibc with INCLUDE_RPC=true (contained in the Config file)
#define BB_FEATURE_NFSMOUNT
//
// Enable support forced filesystem unmounting
// (i.e., in case of an unreachable NFS system).
#define BB_FEATURE_MOUNT_FORCE
//
// Enable support for creation of tar files.
#define BB_FEATURE_TAR_CREATE
//
// Enable support for "--exclude" and "-X" for excluding files
#define BB_FEATURE_TAR_EXCLUDE
//
// Enable support for tar -z option (currently only works for inflating)
```

```
#define BB_FEATURE_TAR_GZIP
//
// Enable reverse sort
#define BB_FEATURE_SORT_REVERSE
//
// Enable uniqe sort
#define BB_FEATURE_SORT_UNIQUE
//
// Enable command line editing in the shell.
// Only relevant if a shell is enabled. On by default.
#define BB_FEATURE_COMMAND_EDITING
//
// Enable tab completion in the shell.  This is now working quite nicely.
// This feature adds a bit over 4k. Only relevant if a shell is enabled.
#define BB_FEATURE_COMMAND_TAB_COMPLETION
//
// Attempts to match usernames in a ~-prefixed path
#define BB_FEATURE_COMMAND_USERNAME_COMPLETION
//
//Allow the shell to invoke all the compiled in BusyBox applets as if they
//were shell builtins.  Nice for staticly linking an emergency rescue shell,
//among other things. Off by default.
// Only relevant if a shell is enabled.
//#define BB_FEATURE_SH_STANDALONE_SHELL
//
//When this is enabled, busybox shell applets can be called using full path
//names.  This causes applets (i.e., most busybox commands) to override
//real commands on the filesystem.  For example, if you run run /bin/cat, it
//will use BusyBox cat even if /bin/cat exists on the filesystem and is _not_
//busybox.  Some systems want this, others do not.  Choose wisely.  :-) This
//only has meaning when BB_FEATURE_SH_STANDALONE_SHELL is enabled.
// Only relevant if a shell is enabled. Off by default.
//#define BB_FEATURE_SH_APPLETS_ALWAYS_WIN
//
// Uncomment this option for a fancy shell prompt that includes the
// current username and hostname.  On systems that don't have usernames
// or hostnames, this can look hideous.
// Only relevant if a shell is enabled.
#define BB_FEATURE_SH_FANCY_PROMPT
//
// Uncomment this option to disable job control.  Job control lets you
// run jobs in the background (which completely useless for is all you
// are doing is running scripts).  Disabing this is bad for interactive
// use, since when you hit ^C in an application, it will also kill the
// shell.  This adds about 2.5k on an x86 system.
//#define BB_FEATURE_ASH_JOB_CONTROL
//
//Turn on extra fbset options
#define BB_FEATURE_FBSET_FANCY
//
//Turn on fbset readmode support
#define BB_FEATURE_FBSET_READMODE
//
```

```
// Support insmod/lsmod/rmmod for post 2.1 kernels
#define BB_FEATURE_NEW_MODULE_INTERFACE
//
// Support insmod/lsmod/rmmod for pre 2.1 kernels
//#define BB_FEATURE_OLD_MODULE_INTERFACE
//
// Support module version checking
#define BB_FEATURE_INSMOD_VERSION_CHECKING
//
// Support for uClinux memory usage optimization, which will load the image
// directly into the kernel memory.  This divides memory requrements by three.
// If you are not running uClinux (i.e., your CPU has an MMU) leave this
// disabled...
//#define BB_FEATURE_INSMOD_LOADINKMEM
//
// Support for Minix filesystem, version 2
#define BB_FEATURE_MINIX2
//
// Enable ifconfig status reporting output -- this feature adds 7k.
#define BB_FEATURE_IFCONFIG_STATUS
//
// Enable ifconfig slip-specific options "keepalive" and "outfill"
#define BB_FEATURE_IFCONFIG_SLIP
//
// Enable ifconfig options "mem_start", "io_addr", and "irq".
#define BB_FEATURE_IFCONFIG_MEMSTART_IOADDR_IRQ
//
// Enable ifconfig option "hw".  Currently works for only with "ether".
#define BB_FEATURE_IFCONFIG_HW
//
// Allows "broadcast +" to set broadcast automatically based on hostaddr
// and netmask, at a cost of about 100 bytes of code (i386).
#define BB_FEATURE_IFCONFIG_BROADCAST_PLUS
//
// Enable busybox --install [-s]
// to create links (or symlinks) for all the commands that are
// compiled into the binary.  (needs /proc filesystem)
//#define BB_FEATURE_INSTALLER
//
// Enable a nifty progress meter in wget (adds just under 2k)
#define BB_FEATURE_WGET_STATUSBAR
//
// Enable HTTP authentication in wget
#define BB_FEATURE_WGET_AUTHENTICATION
//
// Clean up all memory before exiting -- usually not needed
// as the OS can clean up...  Don't enable this unless you
// have a really good reason for cleaning things up manually.
//#define BB_FEATURE_CLEAN_UP
//
// Support for human readable output by ls, du, etc.(example 13k, 23M, 235G)
#define BB_FEATURE_HUMAN_READABLE
//
```

```
// Support for the find -type option.
#define BB_FEATURE_FIND_TYPE
//
// Support for the find -perm option.
#define BB_FEATURE_FIND_PERM
//
// Support for the find -mtime option.
#define BB_FEATURE_FIND_MTIME
//
//// Support for the find -newer option.
#define BB_FEATURE_FIND_NEWER
//
// Support for the -A -B and -C context flags in grep
#define BB_FEATURE_GREP_CONTEXT
//
// Support for the EGREP applet (alias to the grep applet)
#define BB_FEATURE_GREP_EGREP_ALIAS
//
// Tell tftp what commands that should be supported.
#define BB_FEATURE_TFTP_PUT
#define BB_FEATURE_TFTP_GET
//
// features for vi
#define BB_FEATURE_VI_COLON  // ":" colon commands, no "ex" mode
#define BB_FEATURE_VI_YANKMARK  // Yank/Put commands and Mark cmds
#define BB_FEATURE_VI_SEARCH  // search and replace cmds
#define BB_FEATURE_VI_USE_SIGNALS // catch signals
#define BB_FEATURE_VI_DOT_CMD  // remember previous cmd and "." cmd
#define BB_FEATURE_VI_READONLY  // vi -R and "view" mode
#define BB_FEATURE_VI_SETOPTS  // set-able options,  ai ic showmatch
#define BB_FEATURE_VI_SET  // :set
#define BB_FEATURE_VI_WIN_RESIZE // handle window resize
//
// Enable a if you system have setuped locale
//#define BB_LOCALE_SUPPORT
//
// Support for TELNET to pass TERM type to remote host.  Adds 384 bytes.
#define BB_FEATURE_TELNET_TTYPE
//
// Support for devfs.
//#define BB_FEATURE_DEVFS
//
// End of Features List
//
//
//
//
//
//
//------------------------------------------------
// Nothing beyond this point should ever be touched by
// mere mortals so leave this stuff alone.
//
```

```
#include <features.h>
#if defined(__uClinux__)
 #undef BB_RPM2CPIO  /* Uses gz_open(), which uses fork() */
 #undef BB_DPKG_DEB  /* Uses gz_open(), which uses fork() */
 #undef BB_FEATURE_TAR_GZIP /* Uses fork() */
 #undef BB_UPDATE  /* Uses daemon() */
#endif
#if defined BB_ASH || defined BB_HUSH || defined BB_LASH || defined BB_MSH
 #if defined BB_FEATURE_COMMAND_EDITING
  #define BB_CMDEDIT
 #else
  #undef BB_FEATURE_COMMAND_EDITING
  #undef BB_FEATURE_COMMAND_TAB_COMPLETION
  #undef BB_FEATURE_COMMAND_USERNAME_COMPLETION
  #undef BB_FEATURE_SH_FANCY_PROMPT
 #endif
#else
 #undef BB_FEATURE_SH_APPLETS_ALWAYS_WIN
 #undef BB_FEATURE_SH_STANDALONE_SHELL
 #undef BB_FEATURE_SH_FANCY_PROMPT
#endif
//
#if (defined BB_ASH || defined BB_HUSH || defined BB_MSH) && ! defined BB_TEST
 #define BB_TEST
#endif
//
#ifdef BB_KILLALL
 #ifndef BB_KILL
  #define BB_KILL
 #endif
#endif
//
#ifndef BB_INIT
 #undef BB_FEATURE_LINUXRC
#endif
//
#if defined BB_MOUNT && defined BB_FEATURE_NFSMOUNT
 #define BB_NFSMOUNT
#endif
//
#if defined BB_FEATURE_AUTOWIDTH
 #ifndef BB_FEATURE_USE_TERMIOS
  #define BB_FEATURE_USE_TERMIOS
 #endif
#endif
//
#if defined BB_INSMOD || defined BB_LSMOD
 #if ! defined BB_FEATURE_NEW_MODULE_INTERFACE && ! defined BB_FEATURE_OLD_MODULE_INTERFA
  #define BB_FEATURE_NEW_MODULE_INTERFACE
 #endif
#endif
//
#ifdef BB_UNIX2DOS
```

```
 #define BB_DOS2UNIX
#endif
//
#ifdef BB_SYSLOGD
 #if defined BB_FEATURE_IPC_SYSLOG
  #define BB_LOGREAD
 #endif
#endif
//
#if defined BB_ASH && defined BB_FEATURE_SH_IS_ASH
# define shell_main ash_main
#elif defined BB_HUSH && defined BB_FEATURE_SH_IS_HUSH
# define shell_main hush_main
#elif defined BB_LASH && defined BB_FEATURE_SH_IS_LASH
# define shell_main lash_main
#elif defined BB_MSH && defined BB_FEATURE_SH_IS_MSH
# define shell_main msh_main
#endif
```

F. Sobre este documento

Notas

1. El proceso INIT se refiere al “programa” que toma el control después de la ejecución del linuxrc del CDROM. Puede ser el INIT de una distribución en particular o un programa creado a tal efecto (en el caso de metadistros sería el init.sh.
2. Puede encontrar el archivo utilizado en metadistros en su CVS (http://cvs.hispalinux.es/cgi-bin/cvsweb/?cvsroot=metadistros)
3. Esto se ha definido así en el archivo `isolinux.cfg` con la opción “display isolinux.msg”, que podremos ver en la la sección de nombre *`isolinux.cfg`*
4. Para más información sobre estos comandos, visite la sección “What is the DISPLAY File Format? (http://syslinux.zytor.com/faq.php#format)” de la página de syslinux (http://syslinux.zytor.com/)
5. La información de como crear el mapa de teclado se ha obtenido de un mensaje (https://listas.hispalinux.es/pipermail/meta-distros/2002-October/000349.html) enviado por Luis Llorente (mailto:luis.llorente@hispalinux.es) a la lista de correo (https://listas.hispalinux.es/mailman/listinfo/meta-distros) de Metadistros.
6. Viene en el mismo paquete que LILO
7. Viene por defecto en las distribuciones GNU/Linux, y se denominará `es.kmap.gz` o similar.

8. Puede identificar los enlaces simbólicos por las líneas que poseen el símbolo “->”. El nombre del enlace simbólico es el nombre de la izquierda del símbolo mencionado y la ruta a la que apunta está a la derecha.
9. Sólo se han incluido los módulos que al cargarse no producen problemas.
10. El espacio ocupado actualmente por los módulos es de 1,7 Megabytes.
11. El directorio `keymaps` tiene un tamaño de 179 Kilobytes
12. El idioma por defecto es el inglés, por lo que si no se indica nada, este será el que se cargue.