

UNIVERSIDAD DE GRANADA



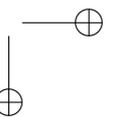
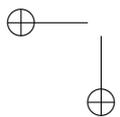
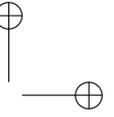
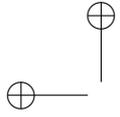
PROYECTO DE FIN DE CARRERA

Technical report nº 3:

Instalación de Linux para ARM con Qemu

Autor:
Ricardo CAÑUELO NAVARRO

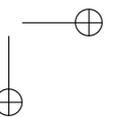
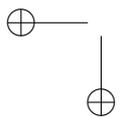
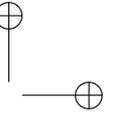
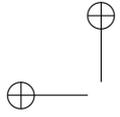
Tutor:
Jesús GONZÁLEZ PEÑALVER



Licencia

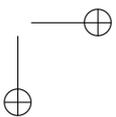
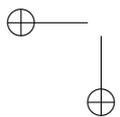
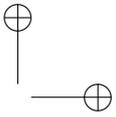
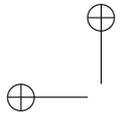
Copyright © 2010 Ricardo Cañuelo Navarro <ricardo.canuelo@gmail.com>.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the file named FDL.



Índice general

1. Introducción	1
2. Motivación	2
3. Herramientas	4
3.1. Software	4
3.2. Hardware	5
4. Instalación básica	6
4.1. Instalación del sistema	6
4.1.1. Preparación	6
4.1.2. Arranque del instalador y particionamiento	8
4.1.3. Instalación del sistema base	13
4.2. ¿Cómo copiar los contenidos del disco duro virtual a la tarjeta?	14
4.3. Pasos finales	15
5. Instalación avanzada	17
5.1. Construcción del kernel a medida	18
5.2. Sistema de archivos a medida	18
5.3. Máquina virtual de Java	18



1

Introducción

En el desarrollo para sistemas empujados, especialmente si trabajamos con herramientas libres que carecen del soporte técnico – y en muchos casos de la documentación – de las herramientas comerciales, nos encontramos en situaciones complicadas que pueden llevarnos a la frustración y hacer que un proyecto que iniciamos con ganas se quede al final sólo en el intento. Con esta documentación intento evitar al lector ese tipo de situaciones (o al menos paliarlas) cuando comience a trabajar en un sistema empujado basado en una BeagleBoard y herramientas libres. Sirve además como documentación de mi propio proyecto, así que si en algunos casos es poco general, se debe a eso.

El primer paso para comenzar la implementación del proyecto es tener un sistema operativo en la placa de desarrollo que nos dé todas las facilidades para manejar sus dispositivos y ejecutar el software que creamos para ella.

Este informe cubre lo necesario para instalar Linux en una placa de desarrollo con procesador ARM como la BeagleBoard. Describiremos también un método para generar software ejecutable por la placa que será el que utilizemos durante el resto del proyecto.

En primer lugar se mostrará una forma sencilla de construir el sistema utilizando el emulador Qemu en nuestra máquina de escritorio de forma que tengamos el sistema básico funcionando rápidamente y sin complicaciones, y más adelante veremos cómo podemos hacer un sistema completo a medida que se adapte mejor a nuestras necesidades.

2

Motivación

Trabajar con una placa de la potencia de la BeagleBoard nos permite alejarnos del tradicional procedimiento de programación de sistemas empotrados y trabajar de una forma más parecida a como lo haríamos en un PC normal. Normalmente para montar un entorno de desarrollo para sistemas empotrados necesitaríamos construir varias *toolchains*, cada una de las cuales probablemente contendría unas bibliotecas distintas que tendrían que casar con versiones determinadas del compilador, enlazador, etc. Esta tarea se acaba convirtiendo muchas veces en un proceso de prueba y error tedioso y largo.

El método que utilizaremos aquí para instalar Linux y las demás herramientas en la BeagleBoard se abstrae por encima del anterior gracias a la potencia de la placa y a la disponibilidad de emuladores en los PCs actuales. Está basado en el howto de Robert Nelson¹, el cual propone dos métodos independientes:

- El primero consiste en cargar en una tarjeta SD una imagen arrancable del instalador del sistema operativo. Así, una vez conectada la placa a un monitor, un teclado y con un acceso a Internet, se arranca la placa con esa tarjeta y se instala Linux igual que haríamos en un PC.
- El segundo realiza la instalación en una máquina virtual en un PC. La instalación es similar a la que se haría en el PC de forma nativa, pero en este caso se instalaría el sistema en la tarjeta SD. Una vez hecha la instalación y configuración ya podemos arrancar la placa con la tarjeta introducida y acceder a ella desde el PC a través del puerto serie.

¹El howto se encuentra en <http://elinux.org/BeagleBoardDebian>

Cada método tiene sus ventajas e inconvenientes. La desventaja del segundo método es que es bastante más lento que el primero, sin embargo será este el método que utilizaremos porque tiene una ventaja importante: que no es necesario tener acceso a la placa durante todo el proceso de instalación y configuración. Podemos hacer todas las tareas y las pruebas necesarias desde el PC y, una vez comprobado que todo está en orden, pasarlo a la placa. Este proceso es más ventajoso especialmente si el trabajo lo realiza un equipo y se dispone de un número pequeño de placas.

Como se ve, en ambas situaciones se hace uso de la capacidad de la BeagleBoard de arrancar desde la tarjeta de memoria. Lo haremos así por ser más sencillo y quizás más adelante instalemos también el sistema operativo en la memoria NAND de la placa como sistema de emergencia.

3

Herramientas

3.1. Software

Hay que tener en cuenta que trabajaremos con herramientas que son *software libre* y que, aunque nos facilitan la vida en muchos aspectos, tienen el inconveniente de que siguen un desarrollo descentralizado, lo que suele ser una fuente de problemas a la hora de encontrar la combinación correcta de versiones entre distintas herramientas que deben cooperar. Por lo tanto aquí se muestran las versiones que se han utilizado porque se sabe que funcionan con seguridad. El lector puede probar versiones más recientes, pero en caso de incompatibilidad entre las herramientas siempre puede recurrir a las que se listan aquí:

- GNU/Linux Debian Lenny o derivada.
- Net Installer de Debian para la arquitectura ARMEL y placa *versatile*:
 - Kernel Linux 2.6.26-2 para ARMEL y placa *versatile*:
<ftp://ftp.us.debian.org/debian/dists/lenny/main/installer-armel/current/images/versatile/netboot/vmlinuz-2.6.26-2-versatile>
 - Disco RAM adjunto:
<ftp://ftp.us.debian.org/debian/dists/lenny/main/installer-armel/current/images/versatile/netboot/initrd.gz>
- Kernel Linux 2.6.29 para la BeagleBoard:
http://rcn-ee.net/deb/kernel/beagle/lenny/v2.6.29-58cf2f1-oer44.1/linux-image-2.6.29-oer44.1.1.0lenny_armel.deb
- Qemu
- uboot-mkimage
- Utilidad para particionar discos (fdisk, cdfisk, gparted, etc)

3.2. Hardware

Para este trabajo se utilizará la BeagleBoard y los componentes necesarios para su conexión. La siguiente lista se muestra a modo de ejemplo y no para que el lector la siga al pie de la letra, ya que los componentes a utilizar varían mucho dependiendo del modo de conexión que se escoja, de la estrategia de instalación y del hardware que ya se tenga.

- BeagleBoard
- Tarjeta SD de 2GB
- Lector de tarjetas SD para el PC
- Conversor serie de 8 pines a DE-9
- Cable serie null-modem con conectores DE-9
- Conversor USB - Puerto serie para el PC
- Cable convertidor de USB tipo A hembra a USB normal hembra
- Hub USB con alimentación externa y puerto *Ethernet*

El montaje y conexionado de estos componentes se escapan a esta documentación. En http://forja.rediris.es/frs/download.php/1637/guia_beagle.pdf hay una guía ligera que describe todo el proceso.

4

Instalación básica

Supondremos que tenemos una conexión a Internet lista y que en el PC estamos utilizando GNU/Linux para realizar todo el proceso. Las instrucciones que se detallarán serán para Debian (en general, cualquier distribución con el mismo sistema de paquetes que Debian), pero se puede utilizar cualquier otra distribución que se desee si sabemos utilizar su sistema de gestión de software para instalar todo lo necesario.

4.1. Instalación del sistema

4.1.1. Preparación

1. En primer lugar instalaremos Qemu si no lo tenemos ya instalado:

```
% apt-get install qemu
```

2. Creamos un directorio de trabajo y bajamos el instalador *netboot* de Debian para la arquitectura ARMEL y el kernel para la placa *versatile*:

```
% mkdir debian-armel
% cd debian-armel
% wget ftp://ftp.us.debian.org/debian/dists/lenny/main/installer-armel/current/images/
  versatile/netboot/initrd.gz
% wget
% ftp://ftp.us.debian.org/debian/dists/lenny/main/installer-armel/current/images/
  versatile/netboot/vmlinuz-2.6.26-2-versatile
```

3. Necesitamos una partición VFAT al principio de la tarjeta SD porque ahí es donde irá instalado el cargador de arranque u-boot. Podemos hacer esta

operación con cualquier programa de particionamiento como cfdisk, fdisk o gparted.

4. Ya podemos empezar la instalación de Debian en la máquina virtual. Si tenemos preparada una tarjeta SD, podemos hacer que Qemu la utilice como disco duro de la máquina virtual y realizar la instalación directamente en ella. Otra opción es crear un disco duro virtual en nuestro disco duro físico y hacer la instalación en él para posteriormente pasar los datos a la tarjeta.

- a) Si utilizamos la tarjeta como disco duro virtual, podemos arrancar Qemu de la siguiente manera:

```
% qemu-system-arm -M versatilepb -kernel vmlinuz-2.6.26-2-versatile -initrd \
initrd.gz -m 256 -hda /dev/sdb -append "root=/dev/ram mem=256M"
```

donde /dev/sdb es el dispositivo que representa a la tarjeta SD en nuestra máquina (puede variar de un PC a otro). Para acceder al dispositivo puede ser necesario tener permisos de root, así que habrá que estar autenticado como tal en el sistema o bien lanzar la orden con sudo si estamos en la lista de sudoers.

- b) Si queremos usar un archivo en nuestro PC como disco duro virtual, tenemos que crearlo. Para ahorrarnos problemas a la hora de pasar los datos a la tarjeta, es importante crear esta imagen de forma que la disposición de las particiones sea análoga a la de la tarjeta de memoria. Es decir, que contenga las mismas particiones, del mismo tipo y con la misma numeración (esto viene a ser: mismo orden). Un poco más adelante veremos cómo realizar el particionamiento utilizando el instalador de Debian.

En primer lugar creamos la imagen con formato *raw* porque así podremos después montarla en el sistema de archivos de nuestro PC para poder acceder a su contenido con facilidad.

```
$ qemu-img create -f raw hd0.img 1G
Formatting 'hd0.img', fmt=raw, size=2097152 kB
```

Una vez creado ya podemos arrancar Qemu indicándole que es ese archivo el que tiene que usar como disco duro:

```
$ qemu-system-arm -M versatilepb -kernel vmlinuz-2.6.26-2-versatile -initrd \
initrd.gz -m 256 -hda hd0.img -append "root=/dev/ram mem=256M"
```

Con esto comenzará la instalación del sistema operativo en la máquina virtual.

4.1.2. Arranque del instalador y particionamiento

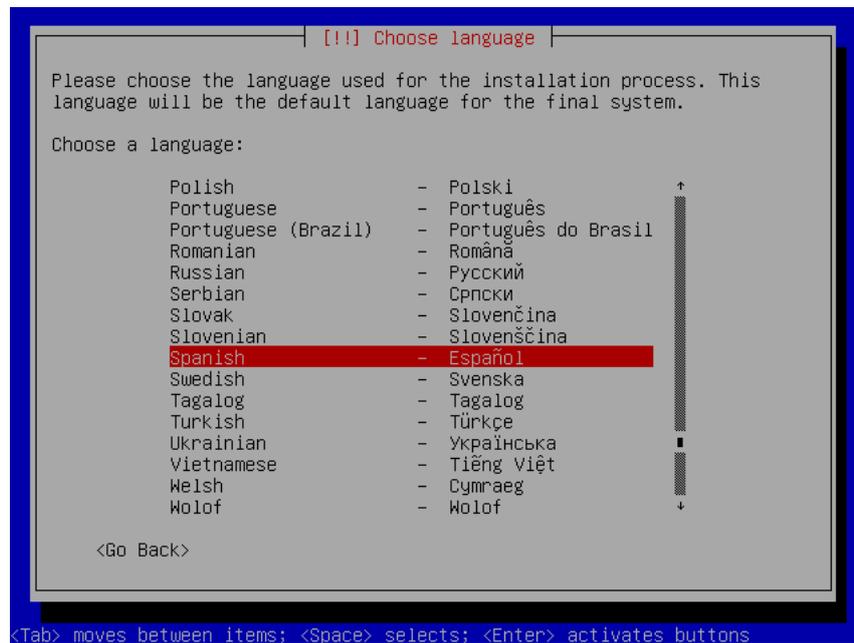


Figura 4.1: Inicio de la instalación

El programa de instalación nos pide que seleccionemos el lenguaje que se utilizará para la instalación y el mapa de teclado a utilizar (seleccionamos español). Después de la detección del hardware y de configurar la red con DHCP nos pregunta el nombre que queremos ponerle a la máquina, así como el nombre del dominio en el que se encuentra. Todo esto no tiene mayor importancia y podemos poner el valor por defecto.

Tras seleccionar el repositorio de Debian más cercano se descargarán e instalarán los paquetes básicos necesarios para llevar a cabo la instalación

El particionamiento es el primer paso importante de la instalación. Para la instalación de Linux es necesario tener al menos dos particiones: una para usar como memoria virtual (partición *swap*) y otra para el sistema de archivos. A su vez el sistema de archivos podemos organizarlo en tantas particiones como queramos, aunque para nuestro sistema nos basta con que esté todo el sistema de archivos raíz en una sola.

Lo más sencillo es realizar el particionamiento desde el mismo instalador de Debian.

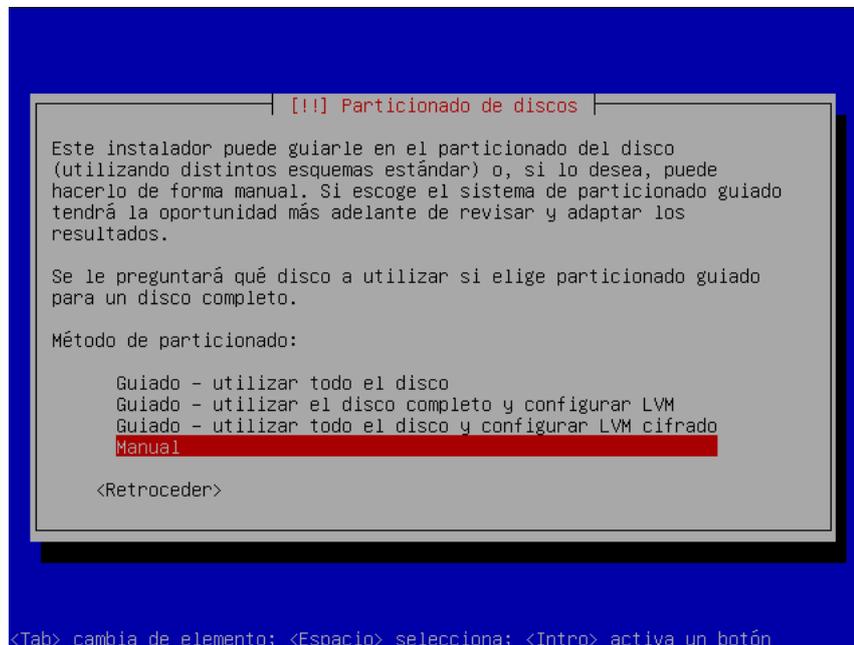


Figura 4.2: Selección del particionado manual

De los tres métodos que nos propone utilizamos el particionado manual (figura 4.2.)

Si estamos utilizando una imagen de disco virtual, ahora podemos crear la primera partición FAT32 para el cargador de arranque. Primero tenemos que crear primero una tabla de particiones para el disco virtual. Para ello lo seleccionamos (figura 4.3) y contestamos “Sí” a la pregunta que nos hace a

continuación el instalador.

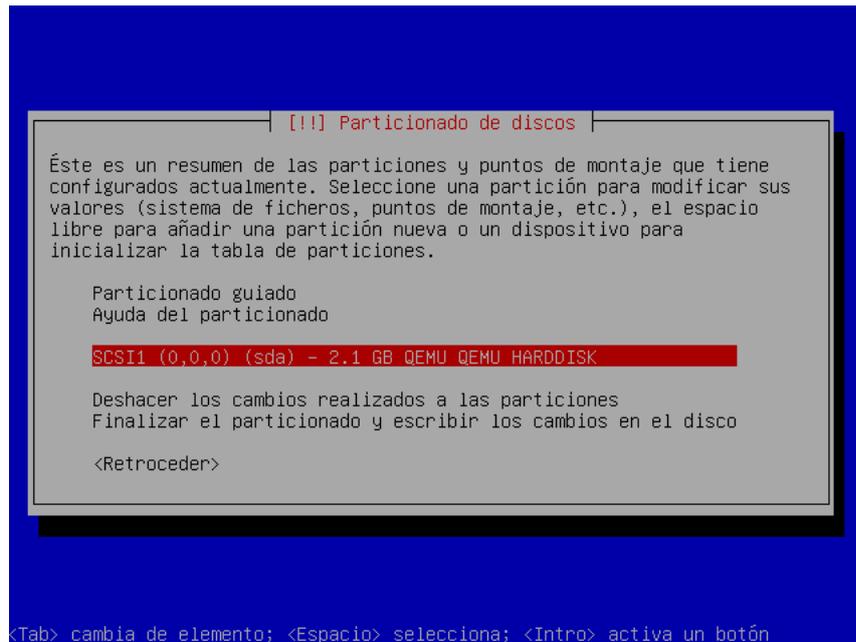


Figura 4.3: Selección del dispositivo a particionar

Ahora que disponemos de una tabla de particiones vacía, seleccionamos la línea marcada como “ESPACIO LIBRE” y elegimos la opción “Crear una partición nueva”. Como tamaño ponemos “50 MB” (sin las comillas) y como tipo “Primaria” y ubicada al principio.

A continuación la configuramos como “sistema de ficheros FAT32” en la opción “Utilizar como”. Como esta partición no tendrá nada que ver con el sistema de archivos raíz de nuestro sistema, elegimos en la opción “Punto de montaje” la opción “No montarla”.

Con esto ya podemos dar por terminada la creación de esta partición.

¿Es realmente necesario realizar esta partición como se detalla aquí? Por supuesto que no. Como casi siempre, hay muchas formas de hacer las cosas. Lo único que tenemos que tener en cuenta es que lo que queremos es “engañar” al sistema emulado en QEMU para que no sepa cuándo se está ejecutando desde

una tarjeta de memoria y cuándo se está ejecutando desde un disco virtual. Así que al final lo único que necesitamos es copiar todos los datos de la partición raíz desde el disco duro virtual a la tarjeta, y nos ahorramos complicaciones si en ambos sitios las particiones se llaman igual, porque así el sistema no notará la diferencia.

Sin embargo, si sabemos manejarnos con la gestión de particiones en linux y sabemos cómo editar el archivo `fstab` para adecuarlo a nuestro sistema, podemos realizar esta parte del particionado como mejor nos convenga.

Las siguientes particiones se hacen de la misma manera tanto si utilizamos la tarjeta SD como si usamos el disco virtual.

Con el espacio libre que queda creamos otra partición que contendrá al sistema de archivos raíz. Seleccionamos para esta partición un tamaño tal que al final del disco quede suficiente para la partición swap, por ejemplo, podríamos coger de tamaño 1960 MB si estamos utilizando un disco de 2 GB. La partición será de tipo “primaria” colocada al principio del espacio libre disponible. El sistema de archivos será “sistema ext3 transaccional”, el punto de montaje “/” (la raíz del sistema de archivos) y tenemos que activar la marca de arranque. En el resto de opciones podemos dejar los valores por defecto.

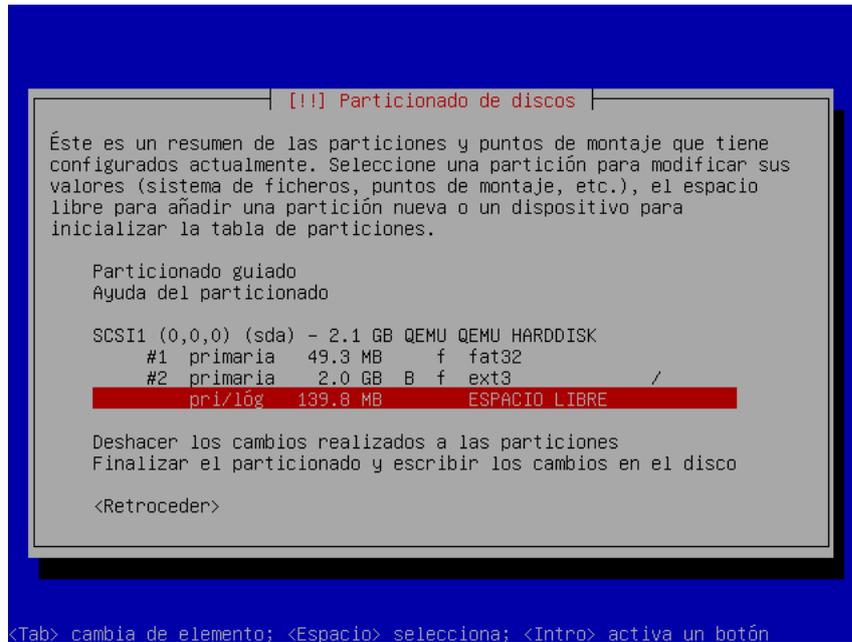
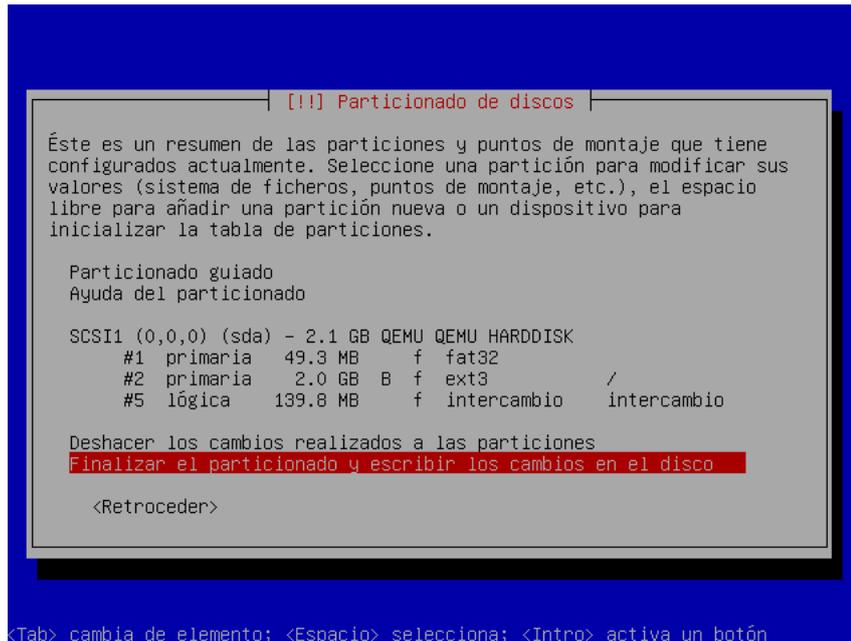


Figura 4.4: Selección del espacio libre

De nuevo seleccionamos el espacio libre que queda (figura 4.4) y lo utilizamos para crear la partición de intercambio o “swap”. Esta será una partición lógica de 148 MB y ubicada al final del disco. En la opción “Utilizar como” seleccionamos “área de intercambio”.

Al final la tabla de particiones quedará así:



```

[!] Particionado de discos

Éste es un resumen de las particiones y puntos de montaje que tiene
configurados actualmente. Seleccione una partición para modificar sus
valores (sistema de ficheros, puntos de montaje, etc.), el espacio
libre para añadir una partición nueva o un dispositivo para
inicializar la tabla de particiones.

Particionado guiado
Ayuda del particionado

SCSI1 (0,0,0) (sda) - 2.1 GB QEMU QEMU HARDDISK
#1 primaria 49.3 MB f fat32
#2 primaria 2.0 GB B f ext3 /
#5 lógica 139.8 MB f intercambio intercambio

Deshacer los cambios realizados a las particiones
Finalizar el particionado y escribir los cambios en el disco

<Retroceder>

<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón

```

Figura 4.5: Tabla de particiones final

Tras el particionamiento comenzará la instalación del sistema base, que llevará unas horas.

4.1.3. Instalación del sistema base

Durante esta fase el instalador se encarga de todo, así que podemos despegarnos un momento del ordenador.

Cuando termine la instalación del sistema base, el programa de instalación realizará una instalación inicial básica. Para ello tenemos que proporcionar una contraseña para el superusuario y crear un usuario normal.

La última fase de la instalación consiste en la instalación de software, sin embargo, el instalador nos informa de que debemos reiniciar el sistema en cuanto termine la instalación para que la lista de dependencias de módulos se cree correctamente. de lo contrario podría resultar imposible cargar algunos módulos. De modo que cuando el instalador nos pregunte qué perfil de aplicaciones instalar, dejaremos marcado sólo el sistema base. Una vez que el instalador finalice, reiniciaremos e instalaremos los paquetes desde el sistema ya arrancado.

4.2. ¿Cómo copiar los contenidos del disco duro virtual a la tarjeta?

Partiendo de que tenemos la tarjeta SD particionada del mismo modo que el disco duro virtual, nuestro objetivo es copiar todos los datos de la partición raíz del disco virtual a la partición equivalente de la tarjeta. Hecho esto ya podremos olvidarnos del disco virtual y trabajar siempre con la tarjeta, tanto en QEMU como en la BeagleBoard.

En primer lugar tenemos que montar la partición raíz del archivo `hd0.img` (recordemos que este archivo contiene un disco entero con varias particiones) en un directorio de nuestro PC. Para ello necesitamos saber a partir de qué byte comienza dicha partición para que `mount` sepa qué es lo que tiene que montar.

Para averiguarlo podemos usar la utilidad `file`. Si ejecutamos `file hd0.img` nos mostrará de qué particiones está compuesta la imagen y, entre otra información, el sector del disco virtual a partir del cual empieza cada partición:

```
$ file hd0.img
hd0.img: x86 boot sector; partition 1: ID=0xb, starthead 1, startsector 63, 96327
sectors; partition 2: ID=0x83, active, starthead 0, startsector 96390, 3823470
sectors; partition 3: ID=0x5, starthead 0, startsector 3919860, 273105 sectors
```

Si tenemos en cuenta que cada sector son 512 bytes y que la segunda partición (la partición raíz) comienza en el sector 96390, podemos calcular que el byte en que comienza es $96390 \times 512 = 49351680$. por lo tanto, la orden para montar la partición en un directorio será:

```
% mount -o loop,offset=49351680 hd0.img directorio
```

Seguramente necesitemos tener permisos de root para hacer esta operación.

También podemos hacer estos cálculos basándonos en la información que nos da la utilidad “fdisk”.

Una vez montada la partición raíz del disco virtual, sólo tenemos que copiar todo su contenido en la partición adecuada de la tarjeta de memoria.

Copiamos también el kernel que utilizaremos cuando arranquemos la placa. Para ello primero nos lo bajamos:

```
$ wget http://rcn-ee.net/deb/kernel/beagle/lenny/v2.6.29-58cf2f1-oer44.1/linux-image-2.6.29-oer44.1_1.0lenny_armel.deb
```

Y a continuación extraemos los contenidos con “dpkg -x archivo.deb directorio” y los copiamos a la tarjeta.

Cuando tengamos copiado todo el sistema de archivos raíz en la partición raíz de la tarjeta, ya podremos utilizarla bajo QEMU para instalar todo el software que necesitamos.

4.3. Pasos finales

Para terminar la instalación del software en el sistema de archivos de la tarjeta de memoria, arrancaremos QEMU con el mismo kernel para armel que utilizamos en la instalación pero con un disco ram distinto. Antes utilizamos uno que contenía el instalador, ahora necesitamos uno apropiado para arrancar nuestro sistema. Además, tenemos que decirle que utilice como disco la tarjeta de memoria y cuál es la partición raíz.

```
% qemu-system-arm -M versatilepb -kernel vmlinuz-2.6.26-2-versatile \  
-initrd initrd.img-2.6.26-2-versatile -m 256 -hda /dev/sdb -append \  
"root=/dev/sda2 mem=256M"
```

Donde /dev/sdb es el dispositivo asociado a la tarjeta de memoria en nuestro equipo. Este nombre puede cambiar de una máquina a otra.

Instalamos los paquetes que necesitamos con apt-get o aptitude.

Una vez que tengamos todo el entorno preparado a nuestro gusto nos queda un último paso para poder cargar el sistema en la placa: convertir el kernel en una imagen arrancable por uboot (el cargador de arranque de la BeagleBoard).

Para ello utilizaremos la utilidad mkimage que proporciona uboot. Podemos obtenerla en debian instalando el paquete uboot-mkimage.

Con la siguiente orden:

```
$ mkimage -A arm -O linux -T kernel -C none -a 0x80008000 -e 0x80008000 \  
-n "Linux" -d imagen_del_kernel punto_de_montaje_vfat
```

Le decimos a mkimage que configure la imagen de un kernel de linux para arm sin compresión, que se encuentra en el archivo “imagen_del_kernel” y que la guarde en el punto de montaje indicado. Además incluirá en la imagen una cabecera con las instrucciones necesarias para que al cargar el kernel, uboot lo copie a la dirección 0x80008000 y comience a ejecutarlo a partir de esa dirección.

Desmontamos la tarjeta y ya podemos arrancar la beagleboard con ella.

5

Instalación avanzada

En el anterior capítulo se explicó una forma básica de instalar y configurar el entorno de desarrollo para poder empezar a funcionar rápidamente. Sin embargo hay muchas cosas que se podrían mejorar. Por ejemplo:

- El kernel que se utilizó es bastante grande, hecho para funcionar en un gran número de dispositivos distintos. Si construimos un kernel hecho a medida para nuestra placa podremos conseguir un mejor rendimiento y un kernel más pequeño.
- El sistema de archivos raíz también es muy grande y contiene muchas aplicaciones que puede que no necesitemos. Podemos construirlo también a medida, con las ventajas de tamaño que ello conlleva.
- Los binarios para la familia armel que provee Debian están compilados para la arquitectura ARMv4T, lo cual proporciona una gran compatibilidad con distintos procesadores, pero a cambio perdemos las prestaciones que ofrecen las nuevas arquitecturas, como la ARMv7-A. Si compilamos los binarios para la arquitectura concreta que estamos utilizando ganaremos en eficiencia.
- La máquina virtual de Java que proporciona Debian es muy grande para nuestras necesidades. Además no realiza optimizaciones a código nativo (no utiliza técnicas de compilación JIT), por lo que ofrece menos rendimiento. Podemos intentar construir una máquina virtual más apropiada para la placa.

En esta parte del documento veremos cómo crear paso a paso: primero un kernel a medida para nuestro sistema, después un sistema de archivos adecuado

a nuestras necesidades y por último una máquina virtual de Java como PhoneME, más apropiada para sistemas empotrados.

5.1. Construcción del kernel a medida

Dejando a un lado los detalles de cómo se construye un kernel a partir del código fuente (ya que esta es la parte “avanzada” de la guía), nos centraremos sobre todo en las partes que difieren entre construir un kernel para nuestro PC y construir uno para otro sistema mediante compilación cruzada.

El primer paso es obtener una *toolchain* compuesta por todas las herramientas que necesitemos para construir software para la arquitectura objetivo (la de la placa).

La construcción de *toolchains* basadas en herramientas libres era, hasta hace poco tiempo, una tarea muy pesada por los motivos que se mencionaron en el capítulo de “Motivación”. Sin embargo aquí veremos dos formas muy sencillas de obtener la *toolchain* que necesitamos.

El método más rápido es utilizar una que ya esté disponible en Internet. Las más apropiadas para nosotros son las publicadas gratuitamente por CodeSourcery

(Por terminar)

5.2. Sistema de archivos a medida

(Por hacer)

5.3. Máquina virtual de Java

(Por hacer)