Capítulo 4
Preparar la tarjeta micro SD

1. Introducción

La Raspberry Pi es un ordenador cómodo y de bajo coste. Sus diseñadores, con el objetivo de simplificar y reducir el precio máximo del producto, eligieron hacer que su ordenador arrancara en uno de los tipos de memoria más extendidos: la tarjeta micro SD.

Después de una explicación detallada del arranque de la Raspberry Pi, este capítulo explica cómo recuperar el sistema operativo después de instalarlo en una tarjeta micro SD, tanto para los principiantes como para los más expertos.

La descarga se realiza en Windows 8. Los usuarios de Linux en modo gráfico lo harán fácilmente en este sistema. La instalación del sistema en la tarjeta micro SD se detalla en Windows 8 y en Debian 7. El sistema operativo elegido es el que recomienda la Fundación Raspberry Pi para los usuarios principiantes: Raspbian.

La sintaxis de los comandos que se presentan está limitada a las opciones utilizadas en este capítulo. Para obtener más información, diríjase al capítulo Usar la línea de comandos, y consulte también el `man` (manual) de cada comando.

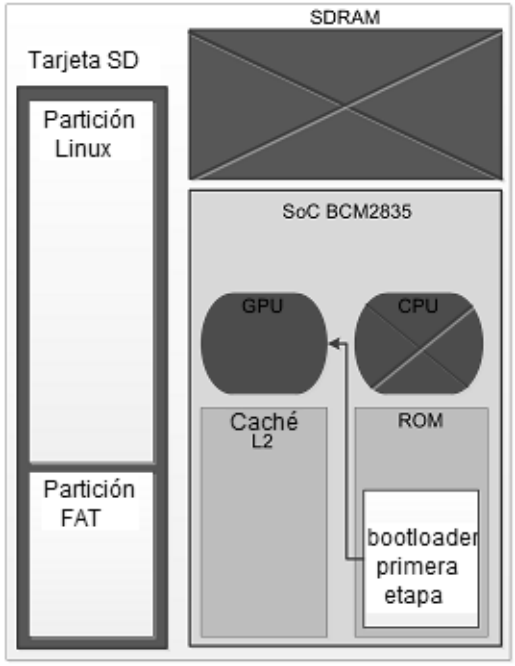
Las versiones de los sistemas operativos y de las herramientas, evolucionan. Tiene que adaptar la siguiente información, en función de las versiones disponibles cuando prepare su tarjeta micro SD.

2. Secuencia de boot de la Raspberry Pi

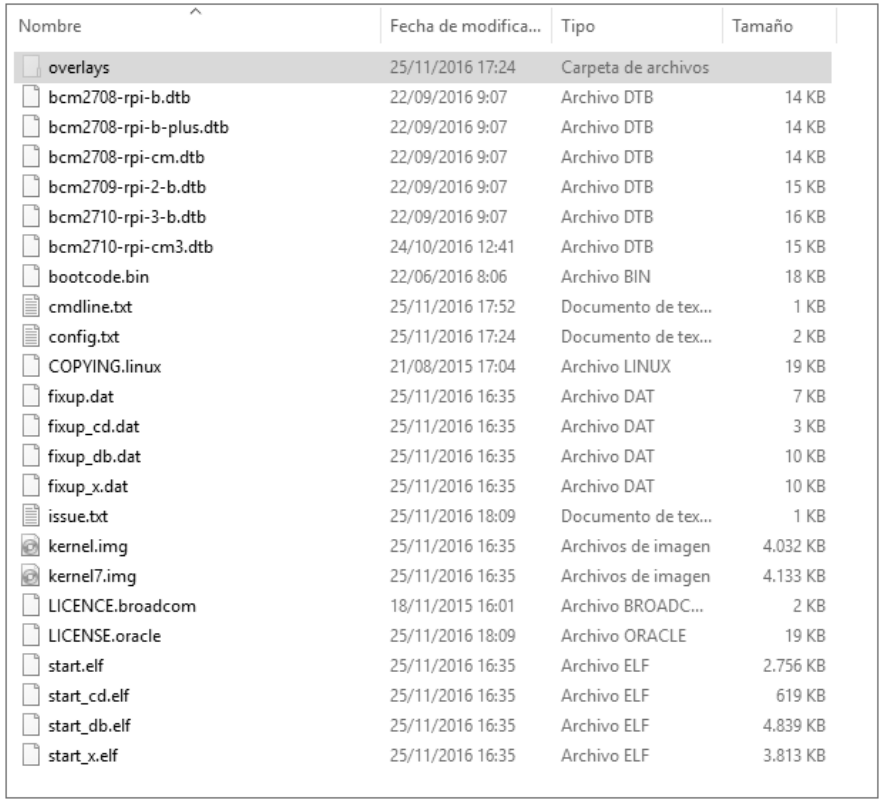
El *boot* (inicio) de la Raspberry Pi es estrictamente idéntico en los diferentes modelos Pi Zero y Rasberry Pi 3. Este proceso de inicio implica un determinado número de operaciones. El correcto entendimiento de esta secuencia es primordial si se intenta modificar voluntariamente el inicio del sistema operativo (iniciar en una llave USB o desde un disco duro). Pero mientras el sistema no arranca, el usuario está en presencia de un enigma que solo el conocimiento del desarrollo exacto de la secuencia de boot permite resolver.

2.1 Etapa 1: enchufar a la corriente

El siguiente esquema representa las diferentes partes implicadas en el inicio de la Raspberry Pi. A la izquierda está la tarjeta micro SD dividida en dos partes. El SoC integra el microprocesador ARM (CPU) y el procesador gráfico (GPU). Está situado a la derecha, bajo la memoria (SDRAM).



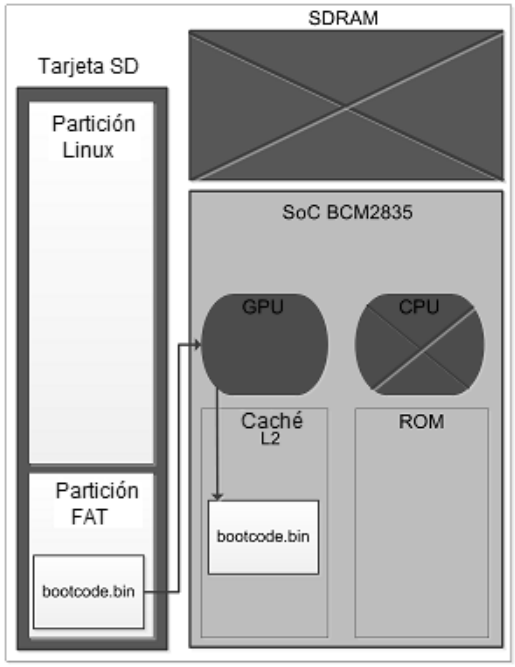
Cuando se enchufa a la corriente, solo está activa la GPU. Mantiene la CPU parada. La memoria SDRAM también está desactivada. El SoC contiene una ROM (*Read Only Memory*), en la que el fabricante ha registrado un programa que forma parte de la cadena de boot. Este programa es el primero de la cadena de boot (*bootloader* = gestor de arranque). No es accesible para el usuario y, por tanto, no se puede modificar. La GPU ejecuta este primer programa, cuyo único papel es acceder a la partición FAT de la tarjeta micro SD para cargar el archivo *bootcode.bin* en memoria. Esta partición contiene los siguientes archivos:



Nombre	Fecha de modifica...	Tipo	Tamaño
overlays	25/11/2016 17:24	Carpeta de archivos	
bcm2708-rpi-b.dtb	22/09/2016 9:07	Archivo DTB	14 KB
bcm2708-rpi-b-plus.dtb	22/09/2016 9:07	Archivo DTB	14 KB
bcm2708-rpi-cm.dtb	22/09/2016 9:07	Archivo DTB	14 KB
bcm2709-rpi-2-b.dtb	22/09/2016 9:07	Archivo DTB	15 KB
bcm2710-rpi-3-b.dtb	22/09/2016 9:07	Archivo DTB	16 KB
bcm2710-rpi-cm3.dtb	24/10/2016 12:41	Archivo DTB	15 KB
bootcode.bin	22/06/2016 8:06	Archivo BIN	18 KB
cmdline.txt	25/11/2016 17:52	Documento de tex...	1 KB
config.txt	25/11/2016 17:24	Documento de tex...	2 KB
COPYING.linux	21/08/2015 17:04	Archivo LINUX	19 KB
fixup.dat	25/11/2016 16:35	Archivo DAT	7 KB
fixup_cd.dat	25/11/2016 16:35	Archivo DAT	3 KB
fixup_db.dat	25/11/2016 16:35	Archivo DAT	10 KB
fixup_x.dat	25/11/2016 16:35	Archivo DAT	10 KB
issue.txt	25/11/2016 18:09	Documento de tex...	1 KB
kernel.img	25/11/2016 16:35	Archivos de imagen	4.032 KB
kernel7.img	25/11/2016 16:35	Archivos de imagen	4.133 KB
LICENCE.broadcom	18/11/2015 16:01	Archivo BROADC...	2 KB
LICENSE.oracle	25/11/2016 18:09	Archivo ORACLE	19 KB
start.elf	25/11/2016 16:35	Archivo ELF	2.756 KB
start_cd.elf	25/11/2016 16:35	Archivo ELF	619 KB
start_db.elf	25/11/2016 16:35	Archivo ELF	4.839 KB
start_x.elf	25/11/2016 16:35	Archivo ELF	3.813 KB

2.2 Etapa 2: carga de bootcode.bin

El siguiente esquema explica la primera fase de inicio de la Raspberry Pi, durante la que GPU carga el código de inicio en memoria.



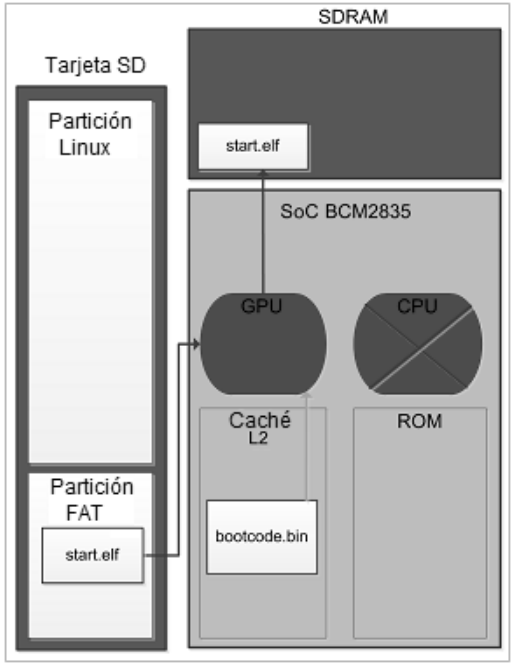
La GPU ejecuta el programa contenido en la ROM del SoC. Accede a una partición FAT de la tarjeta micro SD y encuentra el archivo *bootcode.bin*, que carga en la memoria caché L2 (*Level 2* = memoria caché de nivel 2).

El programa en la ROM, así como el programa *bootcode.bin*, se escriben en código específico para la GPU. Las especificaciones de la GPU no se han publicado, y *bootcode.bin* solo puede distribuirse en formato binario. Este no es un programa libre (licencia Broadcom).

Como la memoria SDRAM montada en el SoC todavía no está activada, es la memoria caché de la GPU, la memoria L2, la que se usa para la carga de *bootcode.bin*.

2.3 Etapa 3: ejecución de bootcode.bin por la GPU

El siguiente esquema explica la segunda fase de inicio de la Raspberry Pi, cuando la GPU carga el firmware (*start.elf*) en memoria.

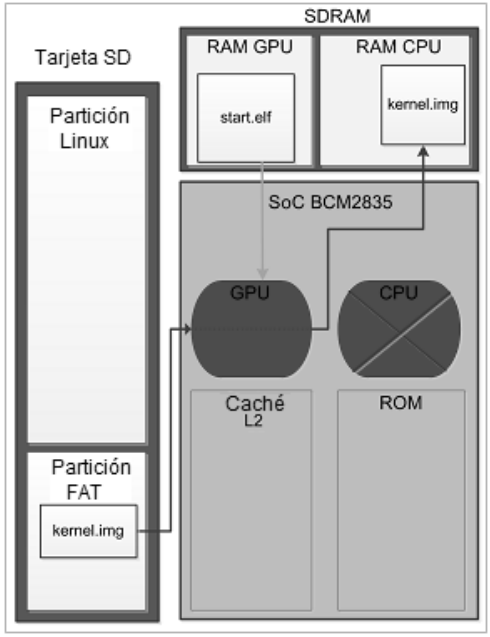


Una vez que *bootcode.bin* está cargado en la memoria caché L2, la GPU ejecuta este programa (flecha gris), que es el segundo nivel del bootloader. El objetivo final es recuperar el programa *start.elf*, que se sitúa también en la tarjeta micro SD, en la partición FAT. *start.elf* es el firmware de la GPU. Como *bootcode.bin*, este programa no es libre y se distribuye en formato binario.

La GPU, bajo las órdenes de *bootcode.bin*, activa la memoria SDRAM de la Raspberry Pi y transfiere una copia de *start.elf* a la parte superior de la memoria. Una vez que *start.elf* está cargado en memoria, *bootcode.bin* le pasa el testigo.

2.4 Etapa 4: ejecución de start.elf por la GPU

El siguiente esquema indica cómo la GPU comparte la memoria con la CPU en función de los argumentos y después carga el núcleo Linux en memoria.



La GPU ejecuta ahora su firmware: *start.elf* (flecha gris).

start.elf reparte la memoria entre la GPU (parte superior) y la CPU ARM (parte inferior), en función del argumento presente en el archivo de configuración (*config.txt*). El argumento *gpu_mem* indica, en MB, la cantidad de memoria que se debe asignar a la GPU. Si este argumento no está presente en el archivo *config.txt*, el valor se fija por defecto a 64 MB. El valor de *gpu_mem* es, como mínimo, igual a 16 MB. Debe ser múltiplo de 16 MB.

Después del establecimiento de las zonas de memoria, el programa *start.elf* transfiere el núcleo Linux *kernel.img* a la parte inferior de la memoria, zona reservada a la CPU ARM. A continuación, lee el archivo *cmdline.txt*, que contiene los argumentos que debe pasar al núcleo durante su ejecución.