Proceso de inicio

1. La BIOS y UEFI

a. BIOS

La **BIOS** (*Basic Input Output System*) es un software de interfaz entre el hardware y el software a un nivel muy básico. Proporciona el conjunto de instrucciones básicas utilizadas por el sistema operativo. Provee el nivel de interfaz más bajo a los drivers y periféricos.

La BIOS está en la memoria **EEPROM** (*Electrical Erasable Programmable Read-Only Memory*) del ordenador. Cuando se enciende el ordenador, o tras un reseteado, se manda una señal llamada *powergood* al microprocesador, que activa la ejecución de la BIOS.

La BIOS efectúa un autochequeo del encendido (POST), luego busca los periféricos, en particular los utilizados para iniciar el sistema. A continuación, se almacena la información relativa al hardware de manera permanente en una pequeña memoria CMOS alimentada por una batería. Al final de proceso, se selecciona el periférico de inicio.

La BIOS lee y ejecuta el primer sector físico del soporte físico de inicio. Suele tratarse de los 512 primeros bytes del primer disco duro (el MBR) o de la partición activa (la PBR) si no hay ningún código presente en el MBR, como se describió en el capítulo Los discos y el sistema de archivos.

b. UEFI

El sucesor de BIOS se llama **UEFI** (*Unified Extensible Firmware Interface*, interfaz extensible de firmware unificada), sucesor de EFI. UEFI se pronuncia como "Unify" pero sin la n. Linux soporta UEFI pero el gestor de arranque debe ajustarse en consecuencia. La mayoría de los PC y placas base vendidos desde el inicio de los años 2010 disponen de UEFI por defecto.

UEFI es un componente software que realiza la interfaz entre el firmware del hardware y el sistema operativo. Garantiza independencia entre el sistema operativo y la plataforma hardware. Su función principal es arrancar el sistema operativo a través de un "boot

manager". Al contrario que BIOS, que busca un programa de arranque en un MBR, UEFI cargará, mediante su microcódigo, todo lo necesario (controladores por ejemplo) para poder iniciar directamente la carga del SO, siempre que éste sea compatible con UEFI.

UEFI dispone de muchas ventajas y funcionalidades adicionales en comparación con BIOS (que ha quedado totalmente obsoleto para los procesadores, controladores y sistemas operativos modernos), tales como:

- una interfaz gráfica;
- soporte para discos de más de 2 TB empleando GPT;
- muchos servicios para gestionar variables (empleando la NVRAM), temporizador (RTC, almacenamiento del huso horario), consolas gráficas, arranque de SO, siendo algunos de estos servicios accesibles desde el SO;
- una selección de diferentes métodos de arranque: UEFI (para arrancar en GPT), CSM (*Compatibility Support Module*, para arrancar desde un MBR), red (PXE, TFTP, DHCP, IP, UDP), y arranque seguro.

Sobre este último punto se ha escrito mucho. El objetivo es impedir que UEFI arranque un sistema operativo o un gestor de arranque que no sea seguro, es decir, que no esté firmado correctamente con una clave verificada; un virus, un rootkit o un SO comprometido no podrán arrancar con un boot seguro.

El problema es que Microsoft anunció que sólo certificaría los equipos que dispusieran de Secure Boot y la clave privada de Microsoft para las versiones superiores o iguales a Windows 8. Por tanto, en estos equipos es imposible arrancar otro SO como Linux, hasta que Microsoft aclare su postura. Muchos fabricantes de Linux han tenido que certificar su sistema de arranque, y por ende comprar una licencia a Microsoft para permitir que su SO arranque en estos equipos. Otros han decidido utilizar métodos alternativos, como el desarrollo de un mini bootloader firmado llamado Shim, un intermediario, compatible con Microsoft, que permite arrancar los SO firmados por sus fabricantes respectivos.

Existen muchos UEFI que pueden emular el comportamiento de un BIOS y pueden continuar arrancando en un MBR clásico, y que ofrecen desactivar Secure Boot permitiendo así que el usuario decida. Aun así, GRUB2 simula un arranque MBR en GPT. Además, la mayoría de los parámetros siguientes funcionan con UEFI.

Observe que UEFI es una norma respetada por la mayoría de los fabricantes, pero algunos lo modifican, lo que puede generar incompatibilidades. Otros, por el contrario, son

estrictos. El fabricante de placas base Gigabyte tiene una gran reputación por la calidad de su implementación UEFI, lo que permite por ejemplo arrancar Mac OS X. Con frecuencia los fabricantes ofrecen una interfaz gráfica o de texto que semejan una BIOS "antigua" y que retoma los ajustes.

En el capítulo anterior, vio algunas generalidades de UEFI en el particionamiento y con respecto a la partición ESP que contiene lo necesario para un arranque del sistema operativo o de un gestor de arranque. En este capítulo, las partes asociadas son comunes a la BIOS y a UEFI, la mayoría de los ajustes que se encuentran en una BIOS también se encontrarán en la configuración de UEFI. Sin embargo, en el caso de que haya detalles suplementarios en UEFI, estos se precisarán.

c. Ajustes básicos

Para no multiplicar UEFI y BIOS, se utilizará la palabra BIOS para referirnos también a UEFI.

Cada BIOS es diferente según los fabricantes de tarjetas madre y los editores (AMIBios, Phoenix, Award, etc.). Para UEFI, si es un estándar, cada constructor propondrá su propia interfaz en modo texto o gráfico. Sea cual sea la marca, muchos de los ajustes son idénticos o, en todo caso, se parecen mucho.

La detección de los discos duros y la elección del soporte de inicio se efectúan desde la BIOS. Linux soporta discos IDE, SATA y SCSI. Es posible que en los antiguos modelos y en las distribuciones Linux antiguas no se reconozca el disco SATA. En este caso, la mayoría de las BIOS tienen como opción emular un IDE modificando el controlador de la SATA. Linux los reconocerá como tales. Utilice este modo solamente como último recurso.

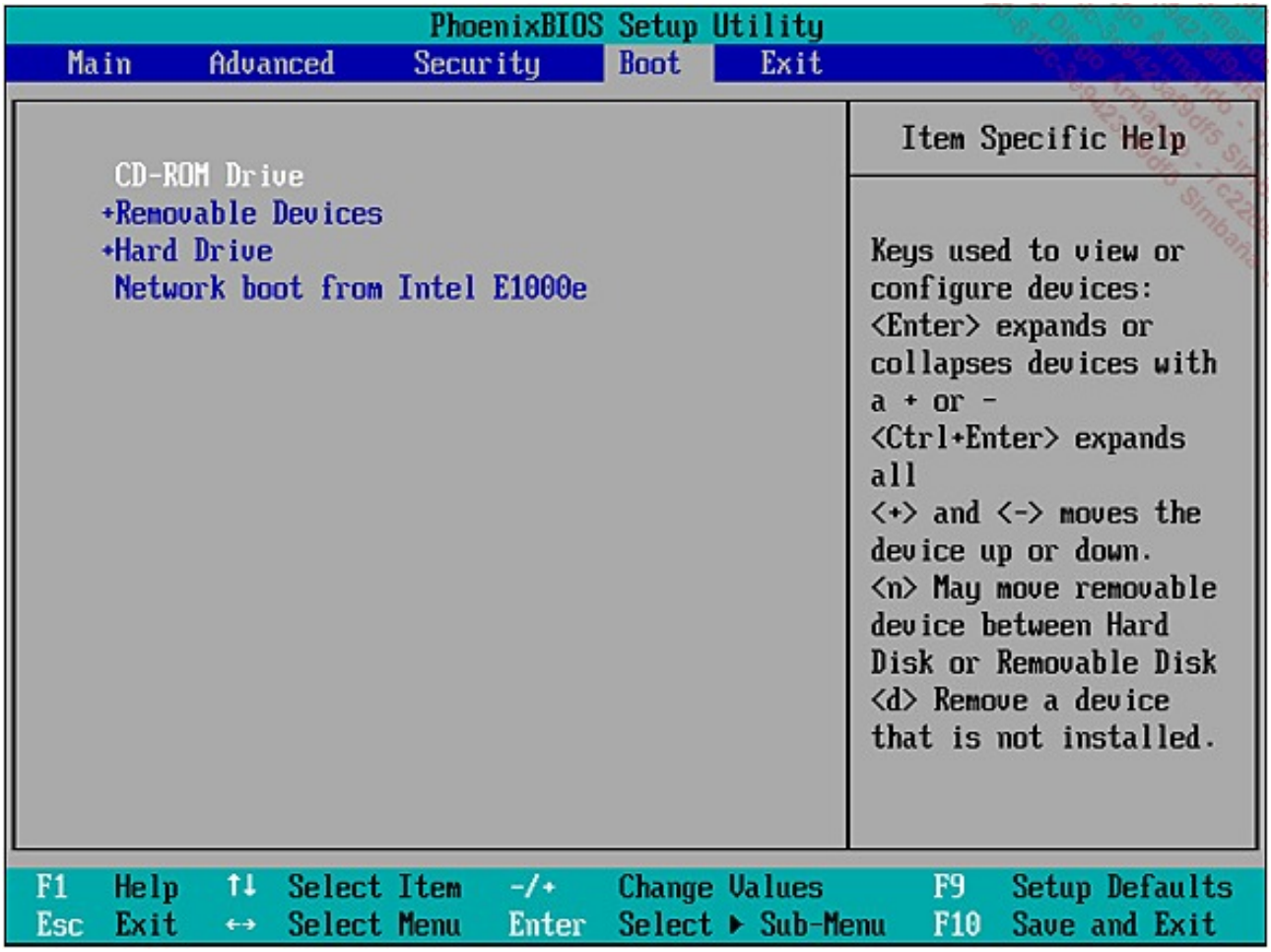
En principio, Linux gestiona correctamente el soporte de los chipsets SATA compatibles **AHCI** (*Advanded Host Controller Interface*), un estándar para las especificaciones públicas que ofrece posibilidades extendidas (NCQ, Trim…). Active esta opción en la BIOS siempre que pueda. En ocasiones aparece con este nombre y, en otras, en modo **nativo**. Si no funciona nada, intente el modo **combined**, **RAID** y luego **legacy IDE**. Encontrará ayuda sobre el SATA en: https://ata.wiki.kernel.org/

Para ejecutar la instalación de Linux desde un soporte óptico (CD-Rom o un DVD-Rom) debe modificar el orden de ejecución de manera que arranque primero desde el lector de CD o DVD, lo que también es válido para los lectores Blu-ray. Esto se efectúa bien en la BIOS misma, o mediante una tecla especial durante el arranque.

Si su teclado es de tipo USB, o sin cable pero con un adaptador sin cable USB, debe activar el **USB legacy support** (a veces esta función se llama **USB DOS function o USB keyboard enable**). Esta funcionalidad permite activar en el momento del inicio el soporte de los teclados, pero también los soportes de almacenamiento (pendrives, discos duros, tarjeta de memoria). Esto no impide que el sistema se encargue del USB: una vez iniciado el SO, los drivers USB del núcleo y de los módulos se encargan del USB. Los modelos de BIOS recientes soportan los teclados en USB por defecto.

Si su teclado es de tipo Bluetooth, no se puede acceder a la BIOS ni a UEFI, excepto con algunos modelos específicos. Observe que los equipos Apple (en los que puede instalar Linux) lo ofrecen desde 2007. Tenga siempre a mano un teclado USB. Los podemos encontrar por 5 euros.

En principio, no tiene por qué tocar la configuración, los experimentos, con gaseosa: no modifique la configuración avanzada del chipset y de otros recursos cuya utilidad no entienda. Sin embargo, con el fin de ahorrar recursos, puede desactivar los puertos de la placa base que no utiliza: puerto paralelo, puerto serie, etc.



Pantalla de BIOS Phoenix para modificar el orden de arranque

El overclocking necesita un hardware especial: procesador, placa base, memoria y alimentación deben ser de alta calidad y el PC debe estar bien aireado. El overclocking es fuente de inestabilidad y cuelgues, tanto en Windows como en Linux. En particular, pone a prueba la memoria, principal causa de inestabilidad, seguida de una alimentación insuficiente o de un procesador más potente. Es la principal causa de inestabilidad. Incluso sin overclocking, resulta útil invertir en componentes de calidad.

2. El gestor de arranque

La BIOS activa el gestor de arranque inicial (*Initial Program Loader*, IPL) a partir de los primeros 512 bytes del soporte de arranque (MBR, PBR). En Linux, el gestor se divide en dos partes. El gestor inicial de los 512 bytes no contiene suficiente código para proponer menús y ejecutar el sistema operativo. Carga un segundo gestor basado en un archivo de configuración.

Este segundo gestor dispone de una interfaz para ejecutar un sistema operativo de entre una selección dada. Puede aprovechar este gestor para pasarle parámetros al núcleo Linux y al proceso init.

La BIOS sólo interviene en el inicio de la máquina, durante la utilización del gestor de arranque y en las primeras etapas de carga del núcleo. Luego, no sirve para nada más. El núcleo dispone de sus propias funciones de detección, aunque se apoye en la configuración de la BIOS. En efecto, ésta, bajo la plataforma Intel, se ejecuta en modo real, y Linux, en modo protegido.

Con GPT, como se vio en el capítulo Los discos y el sistema de archivos, la configuración se encuentra en NVRAM y en la partición ESP. Una variable NVRAM contiene las opciones básicas de arranque, y particularmente el ejecutable que tiene que lanzar para cada entrada. Este binario se encuentra en la partición ESP, llamada más comunmente EFI que puede ser GRUB2.

La mayoría de los UEFI de las cartas modernas son compatibles MBR, o pueden garantizar una compatibilidad MBR con una partición de tipo UEFI. En ese caso, volvemos a un esquema clásico, pero con algunas diferencias si queremos usar GRUB2, que será explicado en la parte correspondiente.

3. GRUB

a. Configuración

En muchas de las distribuciones Linux, como Red Hat 5 y 6 Entreprise, el gestor de arranque por defecto se llama **GRUB** (*Grand Unified Bootloader*). Tienen muchas posibilidades de configuración, en particular la protección con contraseña encriptada.

Cuenta con un intérprete de comandos y con una interfaz gráfica. GRUB guarda las configuraciones en un archivo de texto y no es necesario volver a instalarlo tras cada modificación. GRUB no funciona en modo UEFI, excepto si este último es capaz de gestionar un boot MBR clásico.

Aunque reemplazado por GRUB2 en todas las distribuciones, no piense que GRUB ha desaparecido. Exiten muchas posibilidades, si usted utiliza instancias Amazon EC2 bajo Linux, de que el cargador de arrranque sea la primera versión, ahora llamada "legacy".

A continuación presentamos un ejemplo de configuración a partir del supuesto de que la primera partición del primer disco es /boot y que la segunda tiene una instalación de Windows.

```
timeout=10
default=0
title Red Hat
  root (hd0,0)
  kernel /vmlinuz-2.6.12-15 ro root=LABEL=/
  initrd /initrd-2.6.12-15.img
title Windows XP
  rootnoverify (hd0,1)
  chainloader +1
```

Esta tabla le muestra la sintaxis general de un archivo GRUB:

Parámetro GRUB	Significado
timeout	Número de segundos antes del arranque por defecto.
default n	Arranque por defecto (0=primer título, 1=segundo título, etc.).
gfxmenu	Ruta hacia un menú gráfico.
title xxxx	Principio de una sección, entrada del menú de GRUB.
root(hdx,y)	Se especificarán todos los accesos más abajo a partir de esta partición (ver el significado más adelante). Aquí, hd0,0 representa la primera partición del primer disco detectado por la BIOS. Es la partición /boot.
kernel	El nombre de la imagen del núcleo de Linux, seguido de sus parámetros. La / no indica la raíz del sistema de archivos, sino la de (hd0,0), por lo tanto /boot/vmlinuz...
initrd	Initial ramdisk. Imagen de disco en memoria que contiene la configuración y drivers iniciales y que el sistema sustituirá cuando cargue los discos definitivos.
rootnoverify	La raíz especificada. No debe montarse con GRUB (no soporta NTFS).
chainloader +1	Inicia el primer sector de la raíz especificado más arriba.

Damos el significado de los nombres de periféricos en GRUB.

- (fd0): primer lector de disquetes detectado por la BIOS (/dev/fd0 en Linux).
- (hd0,0): primera partición en el disco duro detectado por la BIOS, ya sea IDE o SCSI (/dev/hda1 o /dev/sda1 según los casos).
- (hd1,4): quinta partición en el segundo disco duro detectado por la BIOS (/dev/hdb5 o /dev/sda5).

b. Instalación

La configuración de **GRUB** reside en `/etc/grub.conf` o `/boot/grub/menu.lst` (el primero es un vínculo al otro). El binario GRUB puede estar instalado en el **MBR** (*Master Boot Record*, los primeros 512 bytes de un disco) o un **PBR** (*Partition Boot Record*, los primeros 512 bytes de una partición).

Para instalar o desinstalar GRUB en caso de que MBR esté corrompido, por ejemplo en `/dev/sda`, utilice el comando **grub-install**:

```
# /sbin/grub-install /dev/sda
```

c. Arranque y edición

Al iniciarse GRUB aparece un menú. Puede ser gráfico o textual, según la configuración. Tiene que elegir una imagen de inicio entre las propuestas con las flechas de dirección. Pulsando la tecla [Entrar], ejecuta la imagen seleccionada.

Puede editar los menús directamente para modificar, por ejemplo, los parámetros pasados al núcleo de Linux o init. En este caso, seleccione una entrada de menú y pulse la tecla e (edit). Aquí se visualizan todas las líneas de la sección. Puede pulsar:

- e: para editar la línea (completarla);
- d: para suprimir la línea;
- o: para añadir una línea;
- b: para iniciar la imagen (booter).

Por ejemplo, para iniciar en modo emergencia (emergency):

- Vaya a la línea Linux o Red Hat y pulse **e**.
- Vaya a la línea kernel y pulse **e**.
- Al final de la línea añada 1 o Single y pulse [Entrar].
- Teclee **b**.

Puede acceder también a un intérprete de comandos pulsando [Esc]. ¡Cuidado! Sólo se reconocen los comandos GRUB.

4. GRUB2

a. GRUB2, el sustituto de GRUB

GRUB2 es el sucesor de GRUB y se ha reescrito desde cero. De su predecesor sólo conserva el nombre. Ya es el gestor de arranque por defecto de todas las distribuciones recientes de Ubuntu desde la versión 9.10 o Debian desde la versión 6. Está dotado de una interfaz gráfica (opcional), y es modular, compatible con varias arquitecturas (BIOS, EFI, Raid, etc.) y dispone de un modo de recuperación. Sin embargo, su configuración es un poco más difícil de comprender.

Algo importante que hay que recordar siempre es que la mayor parte de la configuración es automática. Los componentes de GRUB2 detectarán automáticamente la presencia de nuevos núcleos de Linux y de otros sistemas operativos como Windows, gracias al comando **os-prober**. Esta detección no se realiza en el arranque, sino en la ejecución de un comando de actualización.

Su instalación es idéntica a la de la primera versión de GRUB. Se puede migrar de GRUB a GRUB2 utilizando el comando **upgrade-from-grub-legacy**: el archivo menu.lst pasará a estar controlado y se aplicará la nueva configuración.

b. Configuración

Los archivos de GRUB2 siempre están en **/boot/grub** o **/boot/grub2**, según las distribuciones. Es conveniente adaptar las rutas siguientes según sea el caso. Los nombres de módulos acaban con el sufijo .mod. Hay muchos, especialmente para la

gestión del arranque en un gran número de sistemas de archivos o de tipos de particiones.

La configuración de las entradas de menú está en **/boot/grub/menu.cfg** o **/boot/grub2/grub.cfg**, según la distribución. Sin embargo, no debe modificar este archivo a mano, ya que debe generarse con el comando **update-grub** o **grub2-mkconfig**.

En la ejecución de **grub2-mkconfig**, se analizarán varios archivos para generar la configuración:

- **/bot/grub/device.map**, si existe, para la correspondencia entre los nombres de los discos GRUB y Linux.
- **/etc/default/grub**, que contiene los parámetros por defecto de GRUB.
- Todos los archivos albergados en **/etc/grub.d/**, por orden de lista, que son los scripts que permiten generar automáticamente los menús mostrados en el arranque.

Actualización de GRUB

A cada modificación de estos archivos (o cada vez que se añada un nuevo núcleo o script) debe seguirle la ejecución del comando **update-grub** o **grub2-mkconfig**:

```
# grub2-mkconfig -o /boot/grub/menu.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-514.2.2.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-514.2.2.el7.x86_64.img
Found linux image: /boot/vmlinuz-3.10.0-327.36.1.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-327.36.1.el7.x86_64.img
Found linux image: /boot/vmlinuz-3.10.0-327.28.3.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-327.28.3.el7.x86_64.img
Found linux image: /boot/vmlinuz-3.10.0-327.13.1.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-327.13.1.el7.x86_64.img
Found linux image: /boot/vmlinuz-3.10.0-327.10.1.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-327.10.1.el7.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-c7e8c1200fab4906acacb8590493c6e6
Found initrd image: /boot/initramfs-0-rescue-c7e8c1200fab4906acacb8590493c6e6.img
Found memtest86+ image: /memtest86+.bin
Found Windows 7 (loader) on /dev/sda1
done
```

device.map

El archivo device.map puede editarse manualmente o generarse automáticamente con el comando **grub-mkdevicemap**. Si el archivo device.map no está, la numeración de discos depende del orden de detección de éstos por la BIOS.

Si el comando no está, utilice **grub2-install -recheck**.

```
# grub-mkdevicemap
# cat /boot/grub/device.map
(fd0) /dev/fd0
(hd0) /dev/disk/by-id/ata-STM3500418AS_9VM2LMWK
(hd1) /dev/disk/by-id/ata-WDC_ED5000AACS-00G8B1_WD_WCAUK0742110
(hd2) /dev/disk/by-id/ata-WDC_WD10EADS-00L5B1_WD-WCAU4A320407
```

/etc/default/grub

El archivo **/etc/default/grub** contiene variables que definen las opciones de GRUB2 y opciones por defecto de los núcleos Linux. A continuación se comentan algunas entradas:

Variable GRUB	Significado
GRUB_DEFAULT	Entrada del menú seleccionada por defecto (primera=0).
GRUB_HIDDEN_TIMEOUT	Considerando una duración en segundos, si esta variable está presente, el menú se oculta, lo que permite arrancar en el sistema predeterminado si sólo hay uno. Durante este intervalo de tiempo si el usuario pulsa [Esc] o [Shift] aparecerá el menú por pantalla.
GRUB_TIMEOUT	Duración de la visualización del menú en segundos antes de que arranque la entrada por defecto.
GRUB_HIDDEN_TIMEOUT_QUIET	Define si se visualiza o no el contador de timeout si GRUB_HIDDEN_TIMEOUT está definido: true o false.
GRUB_DISTRIBUTOR	Línea de comandos que define la generación automática del texto en la entrada del menú.
GRUB_CMDLINE_LINUX_DEFAULT	Parámetros pasados al núcleo Linux, únicamente para las entradas “normales”, no las de recuperación.

A continuación un ejemplo:

```
GRUB_DEFAULT=0
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash nomodeset"
GRUB_CMDLINE_LINUX=""
GRUB_GFXMODE=640x480
```

Construcción de los menús

Los menús se construyen con los scripts albergados en **/etc/grub.d/**:

```
# ls -l /etc/grub.d/*
-rwxr-xr-x 1 root root 6658 2011-05-02 21:04 /etc/grub.d/00_header
-rwxr-xr-x 1 root root 5522 2011-04-21 15:19 /etc/grub.d/05_debian_theme
-rwxr-xr-x 1 root root 6291 2011-05-02 21:43 /etc/grub.d/10_linux
-rwxr-xr-x 1 root root 5233 2011-04-21 15:31 /etc/grub.d/20_linux_xen
-rwxr-xr-x 1 root root 1588 2010-09-24 20:16 /etc/grub.d/20_memtest86+
-rwxr-xr-x 1 root root 7119 2011-04-21 15:31 /etc/grub.d/30_os-prober
-rwxr-xr-x 1 root root  214 2010-10-06 14:22 /etc/grub.d/40_custom
-rwxr-xr-x 1 root root   95 2010-10-06 14:22 /etc/grub.d/41_custom
-rw-r--r-- 1 root root  483 2010-10-06 14:22 /etc/grub.d/README
```

Estos archivos se ejecutarán en el orden de listado (orden alfanumérico de forma predeterminada) de ls. Por este motivo empiezan con un valor numérico. Los dos scripts interesantes son **10_linux** y **30_os-prober**.

Si examina el primero, encontrará como elemento particular los bucles que buscan los núcleos Linux en /boot y en la raíz, así como los ramdisk iniciales (ver más adelante), para detectar los distintos núcleos Linux y crear automáticamente las entradas asociadas añadiendo los parámetros por defecto.

En el segundo, verá que se ejecuta el comando **os-prober**, que busca el resto de sistemas operativos en el resto de discos y particiones, como Windows, Hurd, BDS, Mac OS X o incluso Linux. Lo puede comprobar usted mismo:

```
# os-prober
/dev/sda1:Windows 7 (loader):Windows:chain
```

Puede añadir sus propios scripts partiendo de alguna plantilla de las que vienen con el paquete (**40_custom**, por ejemplo). Lo que devuelve su script se añade automáticamente a grub.cfg, del cual hay que respetar su sintaxis.

c. Arranque y edición

El uso de GRUB2 en el arranque es muy parecido al de GRUB, pero con algunas sutilezas. Para editar alguna entrada use siempre la tecla **e** y modifique las distintas líneas como anteriormente. El editor se comporta como un miniEmacs. Se tendrá que usar [Ctrl] **x** para arrancar con sus modificaciones.

Si el menú de GRUB2 no aparece, deje la tecla [Shift] pulsada durante el arranque.

d. Caso de GPT y UEFI

Si su UEFI ofrece un modo de boot Legacy (emulación de BIOS), puede utilizar GRUB2 como gestor de arranque. Deberá crear una partición especial llamada **BIOS Boot partition** con un tamaño mínimo recomendado de 1 MB, identificado con el marcador (flag) **bios_grub** y no formateada. Con gdisk, utilice el código **ef02** como identificador de partición, el marcador se colocará automáticamente, lo que se puede comprobar con parted.

```
# parted /dev/sdb print
...
Number  Start   End     Size    File system  Name          Flags
1      1049kB  2097kB  1049kB               BIOS boot partition  bios_grub
```

GRUB2 es totalmente compatible con el boot UEFI. Si desea emplear este modo, deberá crear una partición EFI (ESP) de tipo **ef00**, montada en **/bott/efi**, de un tamaño de **200 MB**, y con un marcador **boot** y formateada en FAT32 (UEFI deriva de EFI, cuyas primeras versiones datan de 1998, el sistema de archivos más simple era el FAT32). gdisk crea

automáticamente el marcador, la creación del sistema de archivos la realiza usted.

```
# parted /dev/sdb print
...
Number  Start   End     Size    File system     Name            Flags
1       1049kB  211MB   210MB   fat32           EFI System      boot, esp
```

Este es el comando de creación del sistema de archivos FAT32:

```
# mkfs.vfat -F 32 /dev/sdb1
```

Una vez que se inicia el sistema, la partición EFI está accesible generalmente desde el punto de montaje **/boot/efi** y contiene, en el caso de Ubuntu, este tipo de arborescencia:

```
root@ubuntuuefi:/boot/efi# tree
.
+-- EFI
   +-- BOOT
   ¦  +-- BOOTX64.EFI
   ¦  +-- fbx64.efi
   ¦  +-- mmx64.efi
   +-- ubuntu
      +-- BOOTX64.CSV
      +-- grub.cfg
      +-- grubx64.efi
      +-- mmx64.efi
      +-- shimx64.efi
```

En un soporte externo o en el caso de que no se encuentre la NVRAM completa en ese soporte, se ejecutará EFI/BOOT/BOOTX64.EFI por defecto. La arborescencia de Ubuntu contiene todo lo necesario para que arranque GRUB2.

- **grubx64.efi**: el programa de arranque "ordinario"
- **shimx64.efi**: el programa de arranque "Secure Boot"
- **grub.cfg**: la configuración de GRUB2

- **BOOTX64.CSV**: el archivo que será leído en el momento de la instalación para añadir la entrada de boot en NVRAM
- **mmx64.efi**: el ejecutable que firma el Secure Boot con la clave de su máquina y que da el relevo a GRUB.

El comando **efibootmgr** da detalles de los binarios lanzados:

```
# efibootmgr -v|grep -i ubuntu
Boot0004* ubuntu
    HD(1,GPT,ffdd7669-15a0-4500-978e-36c2c5a1b1e6,
0x800,0x100000)/File(\EFI\ubuntu\shimx64.efi)
```

El firmware UEFI ejecutará shimx64.efi, presente en la partición GPT de UUID ffdd-…-b1e6, que corresponde, en la instalación de test, a /dev/sda1, la partición EFI de ese disco.

```
root@ubuntuuefi:/dev/disk/by-partuuid# ls -l ffdd7669-15a0-4500-978e-36c2c5a1b1e6
lrwxrwxrwx 1 root root 10 febr. 1 15:51 ffdd7669-15a0-4500-978e-36c2c5a1b1e6 -> ../../sda1
```

El archivo **grub.cfg** podría ser el mismo que el que se encuentra en /boot/grub, pero podría presentarse de esta forma:

```
search.fs_uuid 19489778-c93b-43a6-9dbf-4790c07cee50 root hd0,gpt2
set prefix=($root)'/boot/grub'
configfile $prefix/grub.cfg
```

O dicho de otra manera, el archivo `/boot/grub/grub.cfg` se cargará. El esquema se encuentra más abajo.

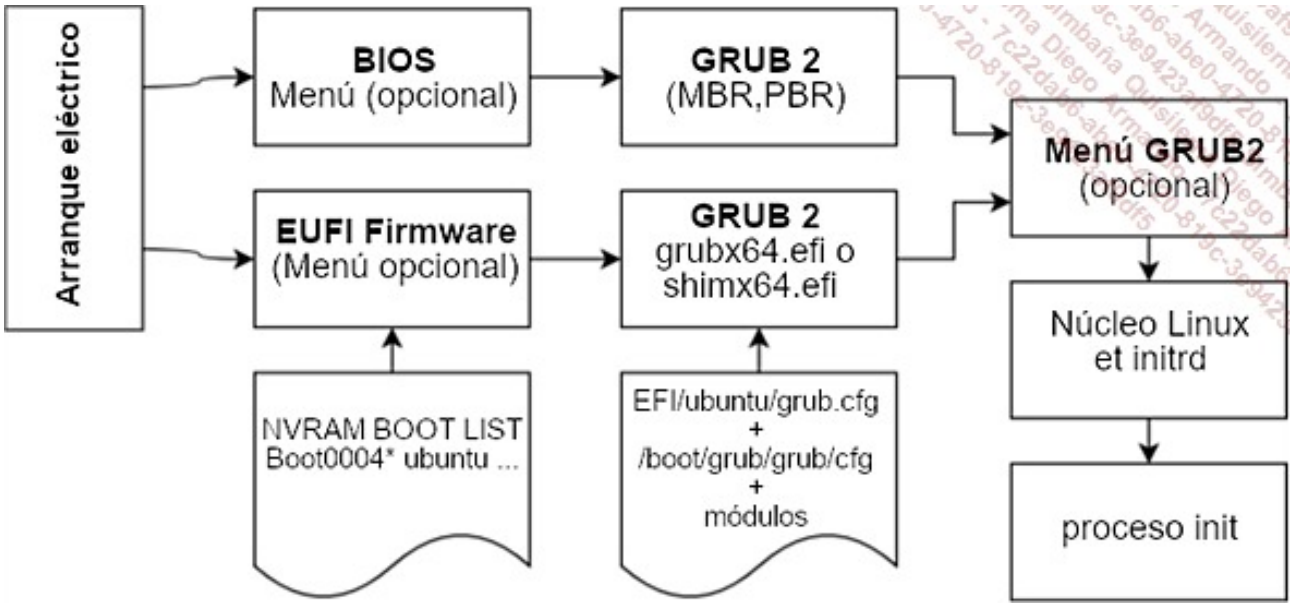
5. Inicialización del núcleo

Durante la carga del núcleo, el monitor le mostrará una tonelada de información. El sistema no la registra en esta etapa, pero en la siguiente, la etapa del init, Linux comienza

a escribir registros en el archivo `/var/log/dmesg` .

- Se detecta e inicializa el hardware.
- initrd está cargado, los módulos presentes se han cargado si es preciso.
- El núcleo monta el sistema de archivos raíz en modo de sólo lectura.
- Crea la primera consola.
- Se ha iniciado el primer proceso (en general init).

El esquema siguiente describe la secuencia de arranque usando BIOS y UEFI:



De la BIOS a o UEFI hasta el arranque de Linux

Existen muchos gestores de arranque, como **LILO** (*Linux Loader*). Sin embargo, GRUB y GRUB2 los han sustituido (salvo en raras excepciones) casi por completo como consecuencia de sus numerosas limitaciones. Otros gestores de arranque como **Clover** permiten también arrancar directamente Linux, u otro gestor de arranque.