Arranque de Linux

Jon Ander Hernández

21 de Mayo de 2012


itsas

¿Por qué es interesante conocer el arranque?

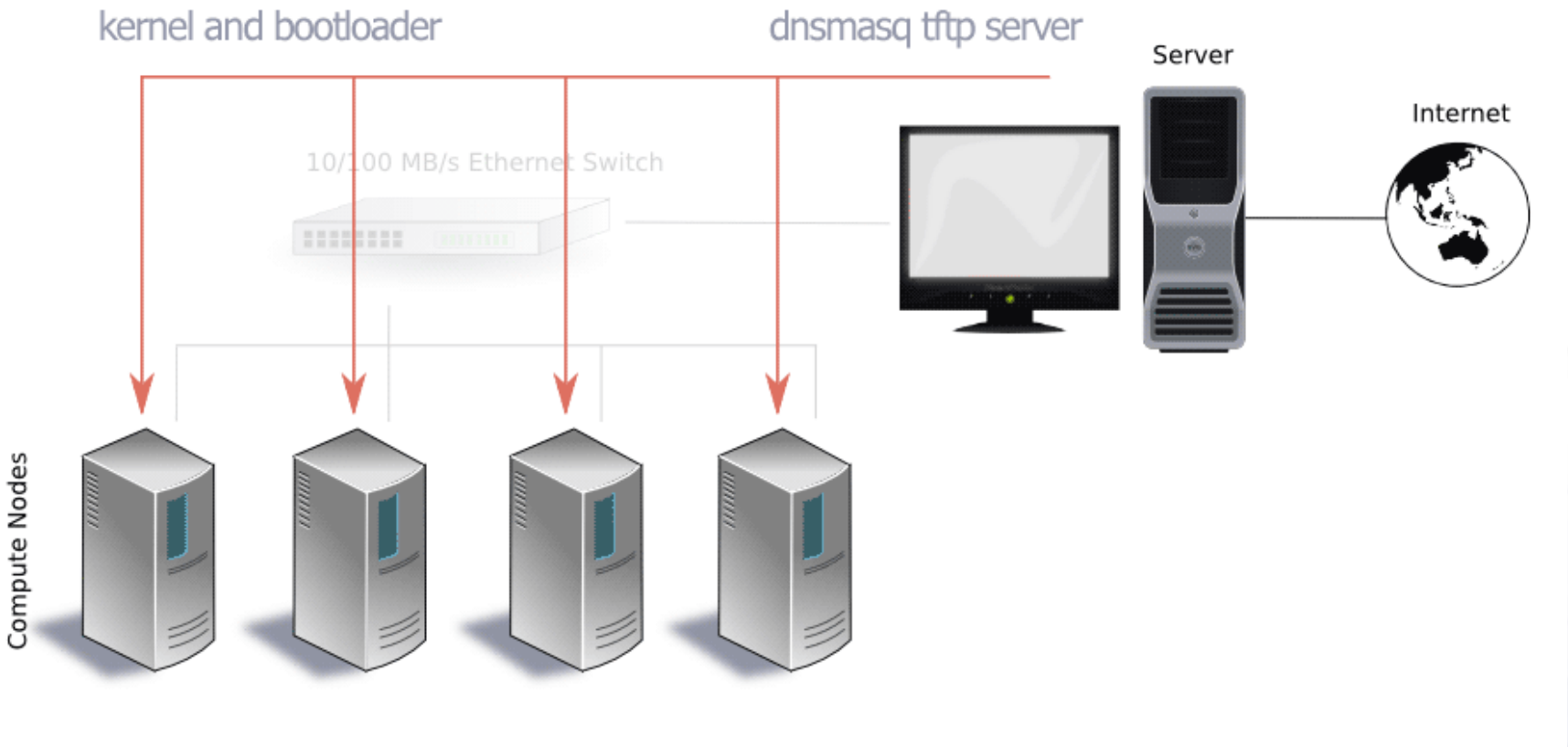
- Para conocer dónde estamos necesitamos entender de dónde venimos.
- Nos ayuda a comprender el funcionamiento y los mecanismos básicos de un sistema operativo.
- Nos ayuda a descubrir los límites, escenarios nuevos de cómo usar un sistema operativo.

Podemos usar Linux en una gran variedad de escenarios y de máquinas.

RAID: Redundant Array of Independent Disks

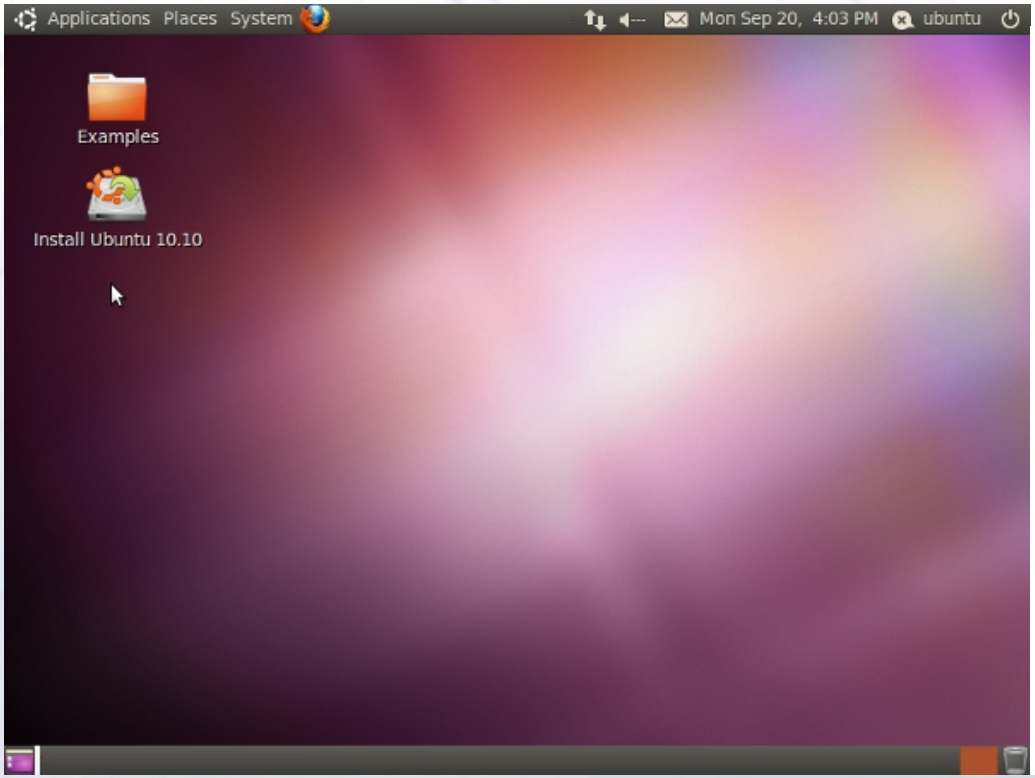
If the root file system appears to be on a software RAID device, there is no way of knowing which devices the RAID volume spans; the standard MD utilities must be invoked to scan all available block devices with a raid signature and bring the required ones online.

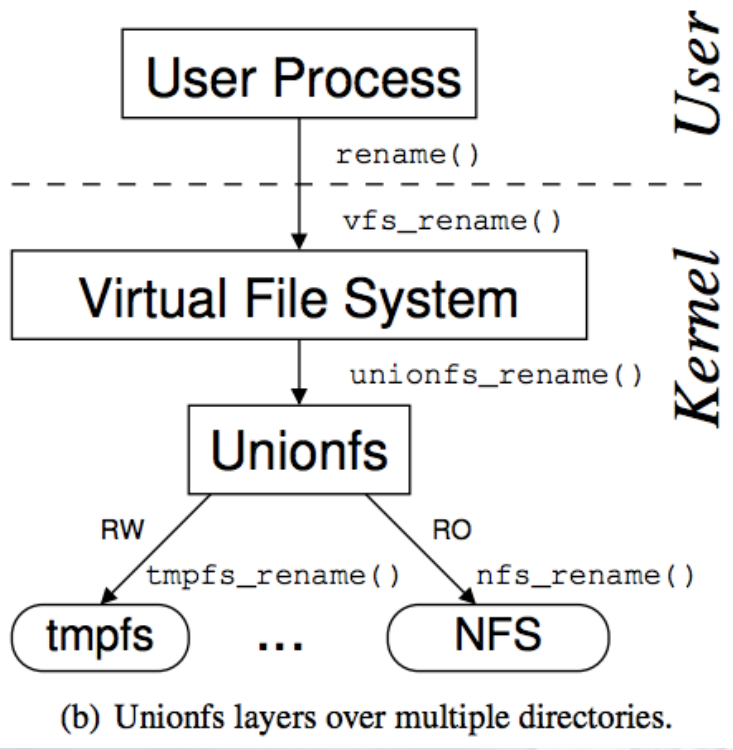
Arrancar por red mediante un NFS



- Bring up the primary network interface.
- Invoke a DHCP client, with which it can obtain a DHCP lease.
- Extract the name of the NFS share and the address of the NFS server from the lease.
- Mount the NFS share.

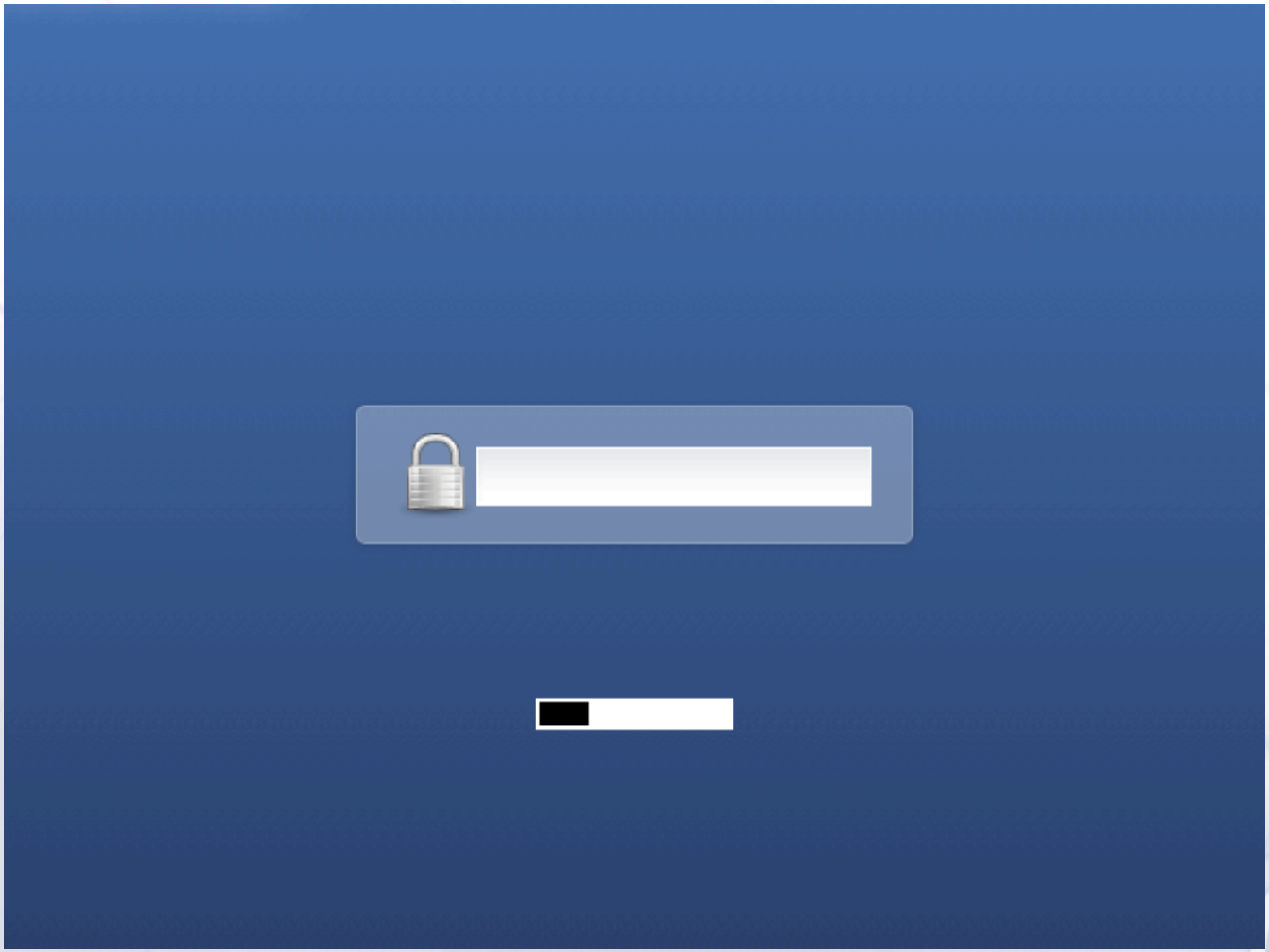
Un sistema de sólo lectura como un liveCD





(b) Unionfs layers over multiple directories.

Volumentes cifrados



- Invoke a helper script to prompt the user to type in a passphrase and/or insert a hardware token.

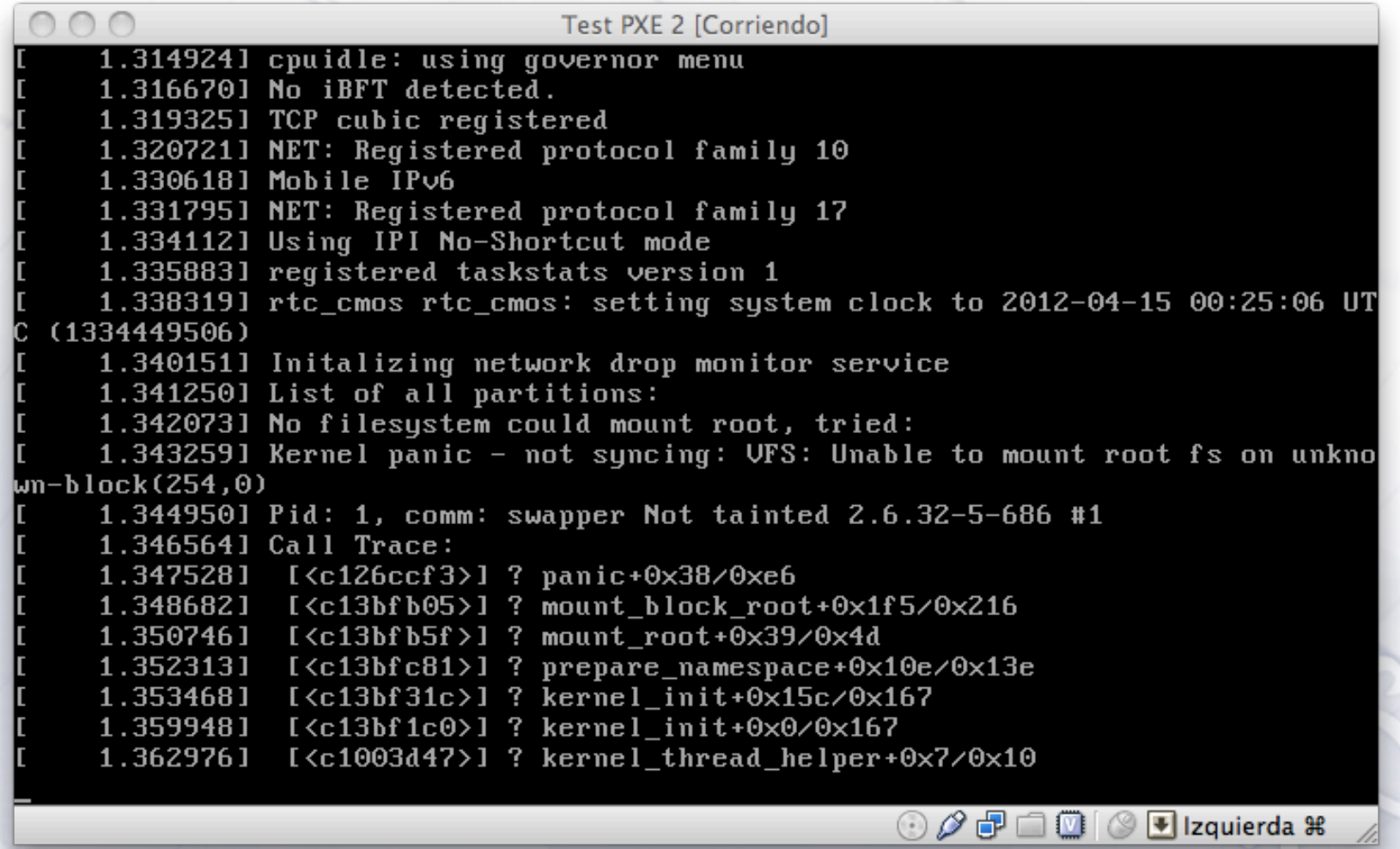
¿Qué es un sistema operativo funcional?

¿Qué es esencial y qué no lo es?

- El kernel, Linux, está cargado en memoria y funcionando.
- Los dispositivos están inicializados y accesibles por las aplicaciones.
- Disponemos acceso a los ficheros del sistema.

Kernel panic - Unable to mount rootfs on unknown-block

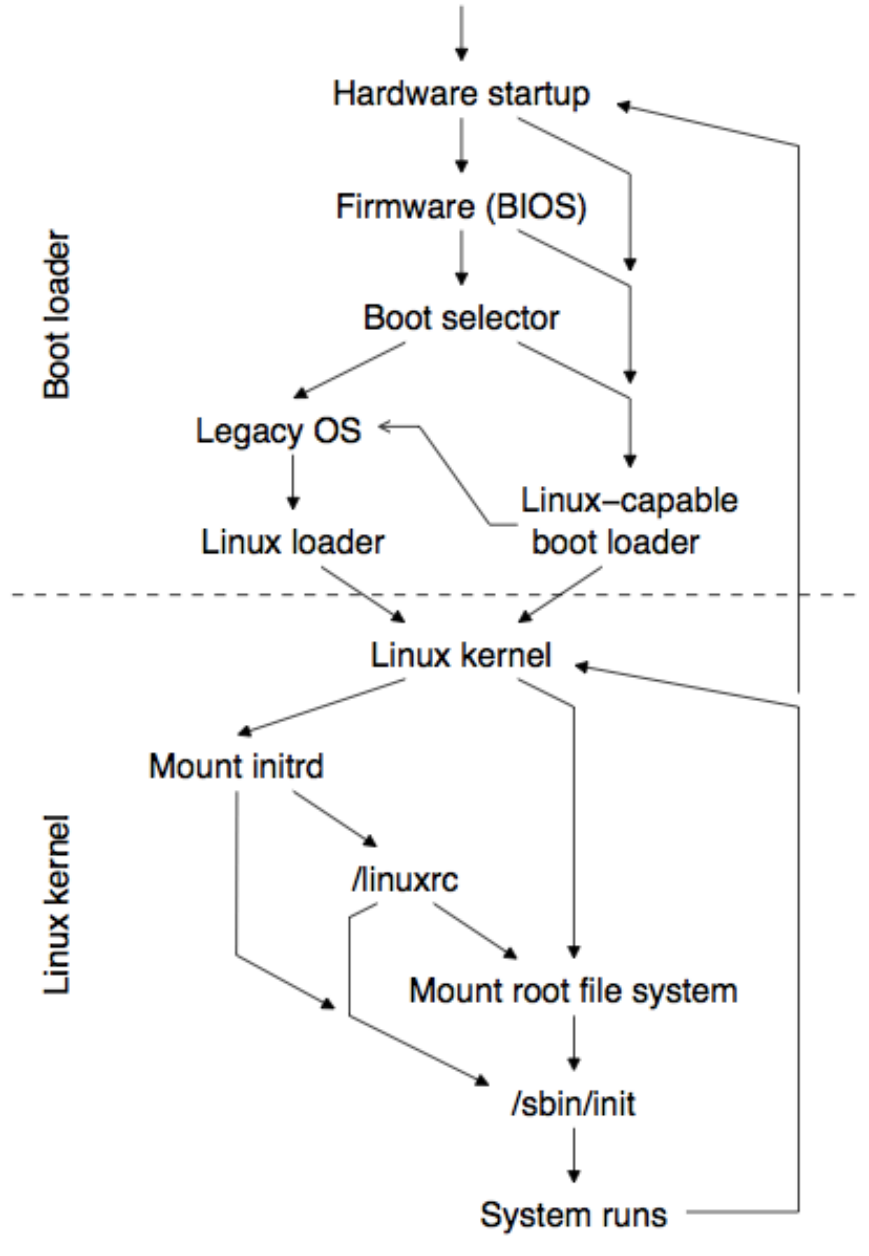
El sistema falla si es incapaz de encontrar un root, dado que sólo con el Kernel el sistema no hace nada.



Un OS funcional puede ser algo muy sencillo y muy pequeño

- Para cargar el kernel necesitamos un **bootloader**.
  - Kernel que normalmente está guardado en la partición /boot
- Para que los dispositivos se inicialicen necesitamos **cargar los módulos** necesarios.
  - Módulos que normalmente están guardados en /lib/modules/$(uname -r)
- Para acceder a los dispositivos del sistema necesitamos **/dev** y por tanto **montar el root**.
  - El kernel debe tener cargados los drivers (o módulos) necesarios para montar el root.

Esquema general del arranque de Linux



bootloaders:

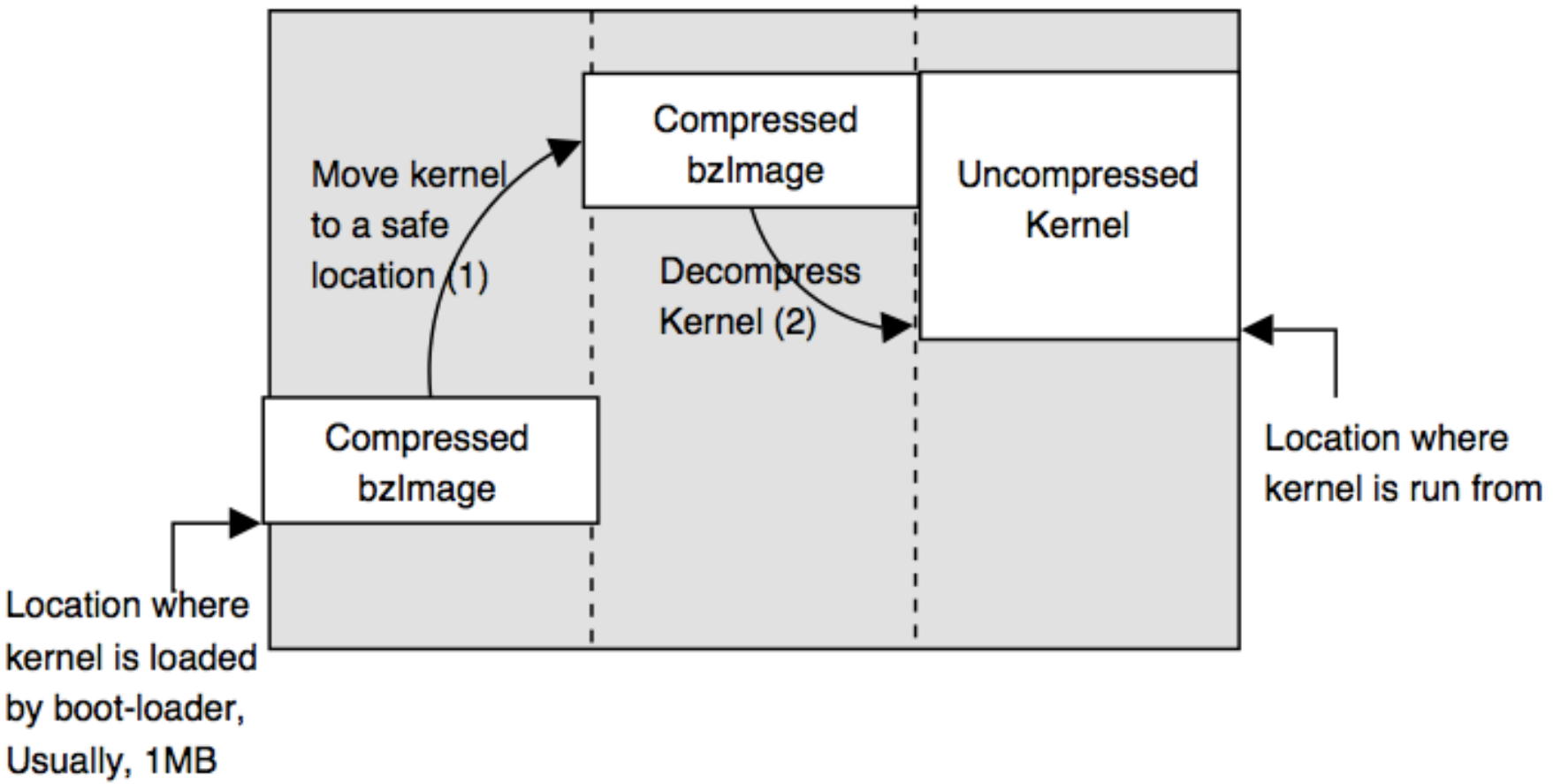
Grub, LiLo, syslinux, kexec

Misión:

Cargar en memoria el kernel Linux.

Pero... dependemos de cada maquina y sus componentes.

El kernel se descomprimirá el sólo al cargarse



Todos los PCs arrancan en modo real como en 1980

Bundesarchiv, B 145 Bild-F077869-0042
Foto: Reineke, Engelbert | 6. April 1988

"The BIOS provides a **small library** of basic input/output functions used to operate and control the peripherals such as"

- the keyboard
- BIOS Enhanced Disk Drive Services (INT 13h)
  logical block addressing (LBA)
- video services (INT 10H)
  setting the video mode, character and string output, and graphics primitives
- PXE

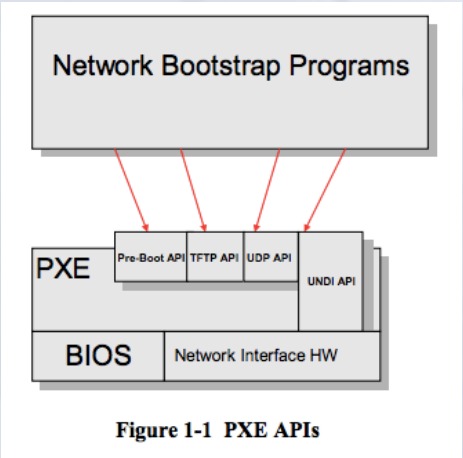


Figure 1-1 PXE APIs

- La BIOS es una librería mínima, mínima.
  - El kernel estará en un sistema de archivos: FAT, NTFS, Ext3+, ISO 9660[1]...
  - Necesitaremos un programa capaz de encontrar y cargar el kernel.
- Algunos dispositivos y/o tarjetas extienden o reemplazan el código de la BIOS para dicho componente:
  - Video BIOS. Establece el modo de video[2].
  - Tarjetas de red.
  - Tarjetas SCSI, RAID, etc...

[1] Compact Disc File System

[2] Para hacer drivers libres es muy frecuente hacer ingeniería inversa sobre la Video BIOS

Filesystem awareness vs filesystem unaware general bootloaders

- LiLo es un bootloader que no necesita leer el sistema de archivos.
  - Puede funcionar con cualquier tipo de sistema de archivos soportado por Linux.
  - Es realmente pequeño.
  - No nos permite hacer gran cosa si el arranque falla.
  - Dependemos de la BIOS para encontrar el disco duro correcto. ¿Y si cambiamos el orden de un disco o lo hace la BIOS?
- Grub 2 es un bootloader modular que hace de todo.
  - Soporta un lenguaje de scripting.
  - Lee bastantes tipo de sistemas de archivos.
  - No soporta nada que requiera un driver (accesos mediante polling).
  - Ocupa bastante.
  - A pesar de ser avanzado sigue siendo muy limitado (¿soporte USB?).

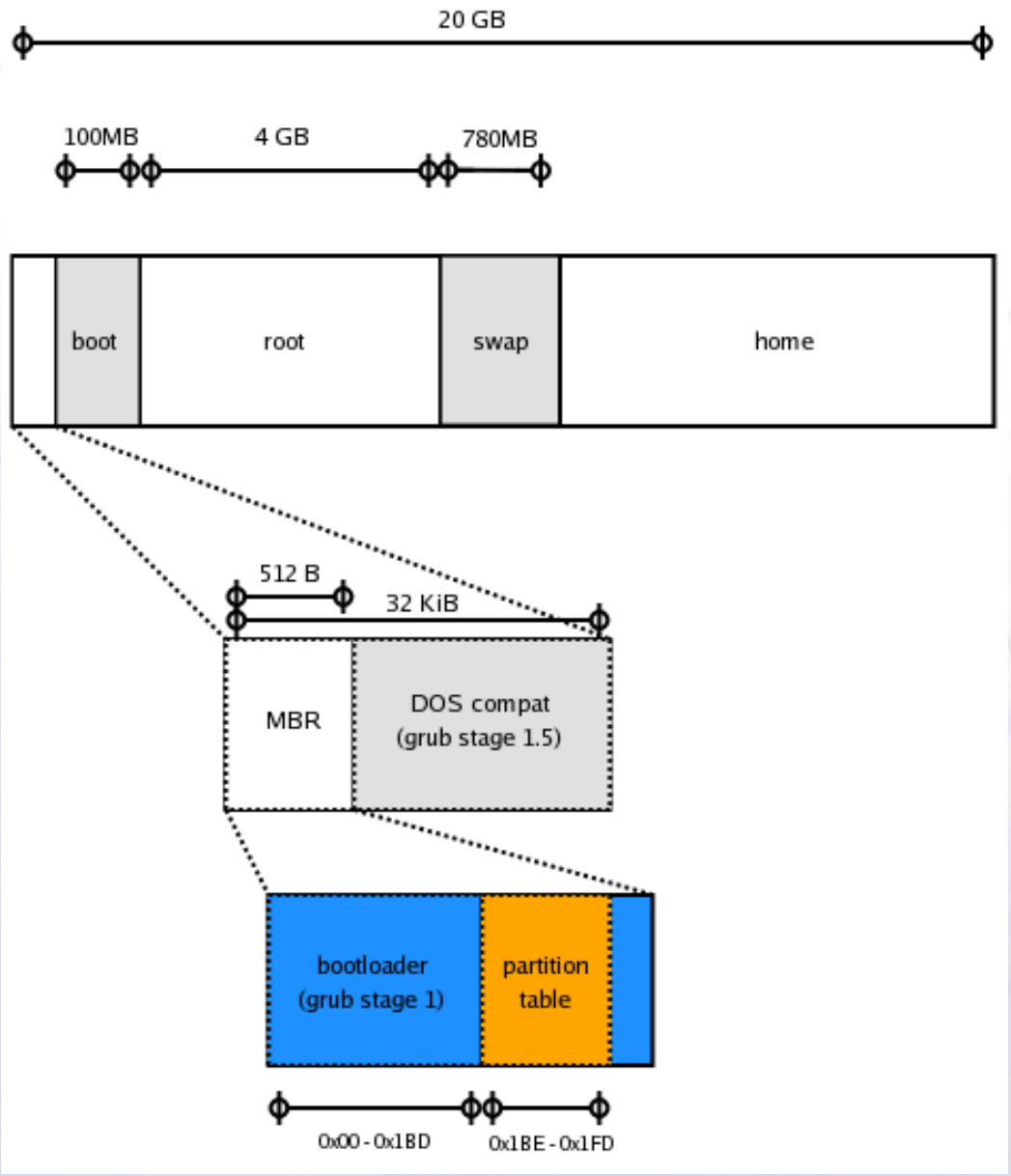
http://repo.or.cz/w/grub2.git/tree/HEAD:/boot/i386/pc boot.S cdboot.S diskboot.S lnxboot.S pxeboot.S

```
# du -hs /usr/lib/grub/i386-pc/*.img
4.0K /usr/lib/grub/i386-pc/boot.img
4.0K /usr/lib/grub/i386-pc/cdboot.img
4.0K /usr/lib/grub/i386-pc/diskboot.img
12K /usr/lib/grub/i386-pc/grldr.img
32K /usr/lib/grub/i386-pc/kernel.img
4.0K /usr/lib/grub/i386-pc/lnxboot.img
4.0K /usr/lib/grub/i386-pc/pxeboot.img
```

```
grub-mkimage --format=x86_64-efi --output=grub.efi --prefix="" part_gpt↩
     part_msdos ntfs ntfscomp hfsplus fat ext2 normal chain boot ↩
    configfile linux multiboot
```

```
grub-mkimage --format=i386-pc-pxe --output=grub.pxe --prefix='(pxe)/↩
    boot/grub' pxe pxecmd
```

En busca de espacio en una partición DOS con Master Boot Record



Boot challenges in the future

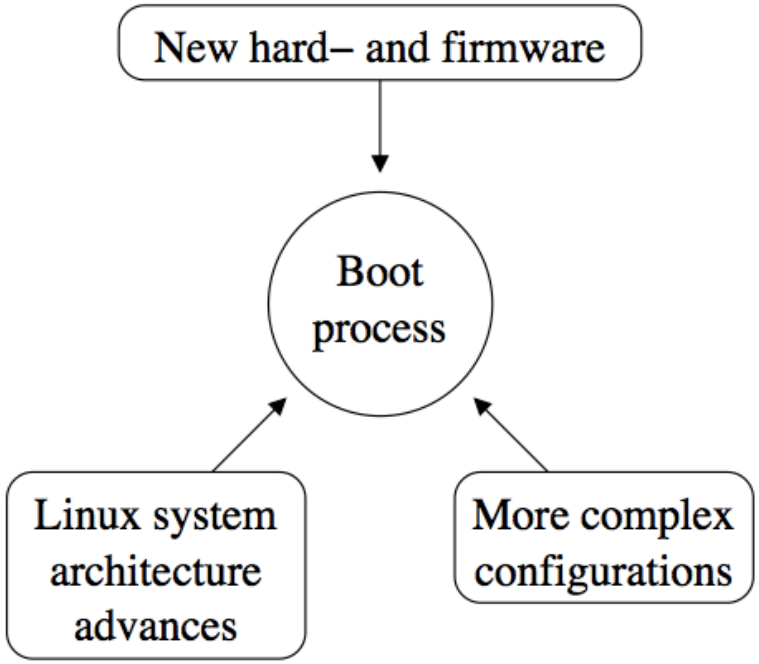


Figure 2: The boot process is facing new challenges from three directions.

El kernel y los dispositivos

Los dispositivos son ficheros

Podemos leer el flujo de comandos PS/2 de un ratón:

```
xxd /dev/input/event2

0000000: 6b50 854f 0000 0000 8125 0200 0000 0000  kP.O.....%......
0000010: 0400 0400 1c00 0000 6b50 854f 0000 0000  ........kP.O....
0000020: 8b25 0200 0000 0000 0100 1c00 0000 0000  .%..............
0000030: 6b50 854f 0000 0000 8b25 0200 0000 0000  kP.O.....%......
0000040: 0000 0000 0000 0000 6d50 854f 0000 0000  ........mP.O....
0000050: cb8f 0000 0000 0000 0400 0400 1e00 0000  ................
0000060: 6d50 854f 0000 0000 d58f 0000 0000 0000  mP.O............
```

major and minor numbers of special files

La asociación entre un driver y un fichero de dispositivo se realiza mediante los *major and minor numbers*.

```
brw-rw---- 1 root disk  8,   0 2012-04-10 17:48 /dev/sda
brw-rw---- 1 root disk  8,   1 2012-04-10 17:48 /dev/sda1
crw-rw---- 1 root video 226, 0 2012-04-10 17:48 /dev/dri/card0
crw-r----- 1 root root  13,  32 2012-04-10 17:48 /dev/input/mouse0
```

Creación manual de un fichero de dispositivo:

```
$ mknod b 8 1 /dev/sda1
```

El problema es que son pocos números para tantos drivers en tantas arquitecturas diferentes.

→ Lo ideal es la:

- *asignación dinámica* de los números por el kernel. Algunos se mantienen fijos por compatibilidad.
- *gestión automática del directorio /dev*. Los ficheros se crean cuando se conecte el dispositivo.

El kernel y los módulos

¿Cuánto ocupa un kernel convencional?

```
$ du -hs /boot/vmlinuz-3.0.0-12-generic
4.5M
```

¿Cuánto ocupan sus los módulos?

```
$ du -hs /lib/modules/3.0.0-12-generic/
105M
```

¿Cuántos módulos hay?

```
$ find /lib/modules/3.0.0-12-generic/ -type f -iname '*.ko' | wc -l
3432
```

¿Cuántos módulos puedo estar usando en un ordenador convencional?

```
$ lsmod | wc -l
78
```

¿Cómo sabe el kernel que hardware soporta un módulo?

- Los módulos expresan el harware que soportan con los modalias, que actúan como expresión regular identificando el dispositivo por el fabricante, número de serie, etc...

```
alias:          pci:v00001002d00009715sv*sd*bc*sc*i*
```

- Los módulos tienen dependencias. La carga de uno conlleva la carga de sus dependencias.

```
depends:        drm,drm_kms_helper,ttm,i2c-core,i2c-algo-bit
```

- Admiten parámetros al cargar el módulo

```
parm:           modeset:Disable/Enable modesetting (int)
```

```
$ modprobe -i radeon modeset=1 agpmode=2
```

On en el arranque

```
linux /vmlinuz radeon.modeset=1 radeon.agpmode=2
```

- Pueden necesitar un firmware (binary blob) para ser operativos.

```
firmware:       radeon/RS600_cp.bin
```

Módulos del kernel

Linux permite que sus drivers estén integrados en el propio binario del kernel o como módulos cargables.

Antiguamente la gente recompilaba el kernel para hacerlo lo más liviano y pequeño posible. Hoy día no merece la pena.

```
$ modinfo radeon
filename:       /lib/modules/2.6.35-22-generic/kernel/drivers/gpu/drm/↩
    radeon/radeon.ko
license:        GPL and additional rights
description:    ATI Radeon
author:         Gareth Hughes, Keith Whitwell, others.
firmware:       radeon/R520_cp.bin
firmware:       radeon/RS600_cp.bin
...
alias:          pci:v00001002d00009715sv*sd*bc*sc*i*
alias:          pci:v00001002d00009714sv*sd*bc*sc*i*
alias:          pci:v00001002d00009713sv*sd*bc*sc*i*
alias:          pci:v00001002d00009712sv*sd*bc*sc*i*
alias:          pci:v00001002d00009711sv*sd*bc*sc*i*
...
depends:        drm,drm_kms_helper,ttm,i2c-core,i2c-algo-bit
parm:           modeset:Disable/Enable modesetting (int)
parm:           agpmode:AGP Mode (-1 == PCI) (int)
...
```

Podemos observar la jerarquía de subsistemas:

- mouse → input layer → usb → pci

```
$ tree /sys/devices/pci0000:00/0000:00:06.0/usb2/2-1/2-1:1.0/input/↩
    input6/mouse0
 |-- dev
 |-- device -> ../../input6
 |-- power
 |   |-- async
 |   |-- autosuspend_delay_ms
 |   |-- control
 |   |-- runtime_active_kids
 |   |-- runtime_active_time
 |   |-- runtime_enabled
 |   |-- runtime_status
 |   |-- runtime_suspended_time
 |   `-- runtime_usage
 |-- subsystem -> ../../../../../../../../class/input
 `-- uevent
```

El fichero uevent permite regenerar los eventos de hotplug [3].

[3] El coldplugin se utiliza para regenerar los eventos simulando que se han vuelto a añadir, porque dichos eventos se produjeron durante una fase muy temprana del arranque cuando el sistema no había arrancado y las aplicaciones que tenían interés en tales dispositivos aún no estaban listas.

udev. A Userspace Implementation of devfs

- Hotplugging. Cuando conectamos un dispositivo queremos que se configure automáticamente.
  Ejemplos:
  - Si enchufamos un pendrive queremos que se monte automáticamente.
  - Si enchufamos una cámara de fotos queremos que arranque el gestor de fotografías.
  - Si enchufamos una sintonizadora de TDT queremos que se cargue el módulo del kernel y cargue el firmware (binary blob) en el dispositivo.
  - Cuando enchufamos unos altavoces USB queremos que empiecen a sonar inmediatamente.

- Queremos persistencia en los nombres de los dispositivos.
  - Si cambiamos de puerto USB queremos que el dispositivo tenga siempre el mismo nombre. Como por ejemplo el nombre de un interfaz de red wifi (wlan0, wlan1, etc..).
  - El disco duro siempre se tiene que poder localizarindependiente de dónde y cómo esté conectado. /dev/sda2 no es una opción.

- Los eventos de hotplug se obtienen automáticamente a través de sysfs. Y se emplea la información accesible desde sysfs.
- Funciona mediante un sistema de reglas, que permite definir *la política de nombres* fuera del kernel[4].

udev simplifica la carga de módulos y el firmware de una manera tremendamente elegante:

`lib/udev/rules.d/80-drivers.rules`:

```
DRIVER!="?*", ENV{MODALIAS}=="?*",
    RUN+="/sbin/modprobe -bv $env{MODALIAS}"
```

`/lib/udev/rules.d/50-firmware.rules`:

```
SUBSYSTEM=="firmware", ACTION=="add",
    RUN+="firmware --firmware=$env{FIRMWARE}
                   --devpath=$env{DEVPATH}"
```

[4] Los BSD usan DevFS, pero fue rechazado en Linux. Recientemente existe una versión mínima llamada devtmptfs destinada a sistemas empotrados.

Persistencia de los nombres de los dispositivos

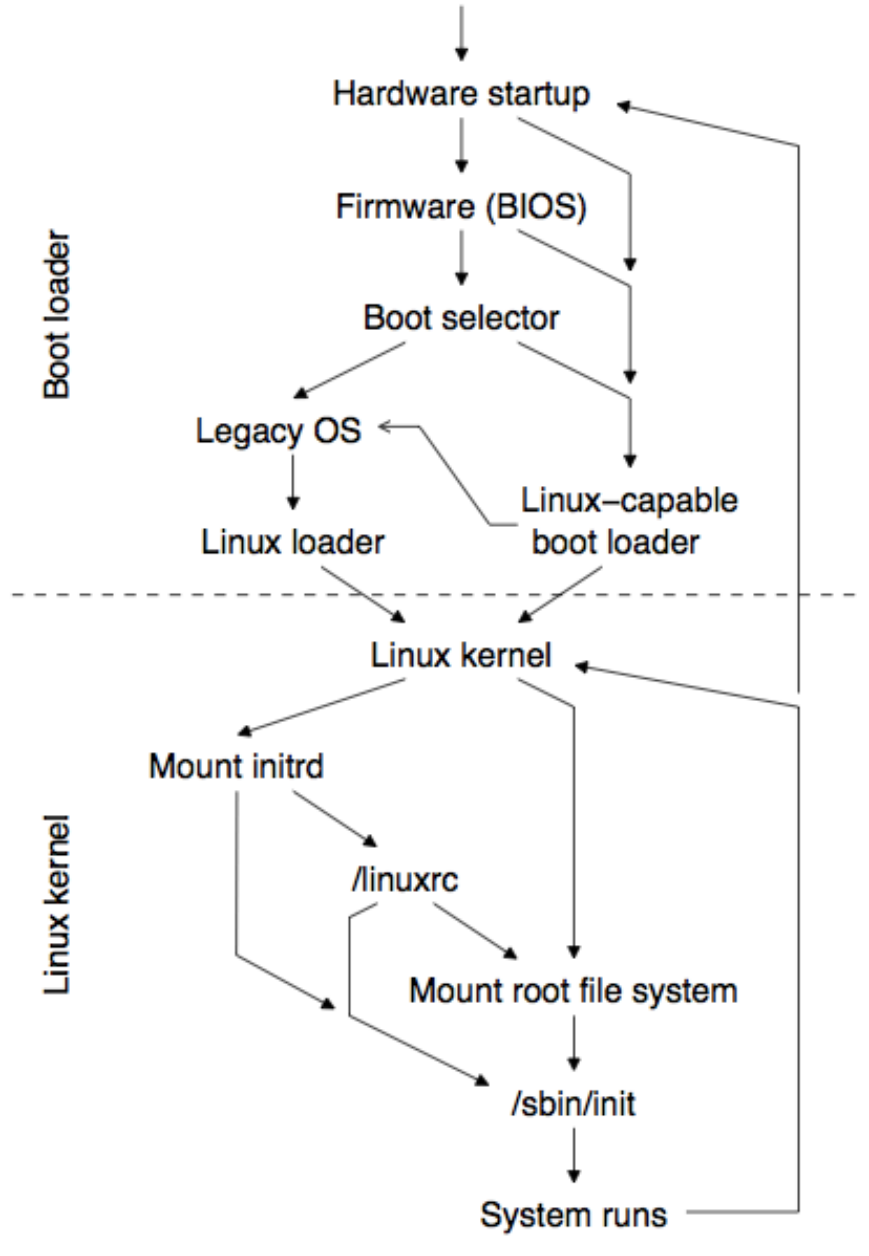
```
$ tree /dev/disk/
 |-- by-id
 |    |-- ata-VBOX_HARDDISK_VB946693c0-53b88d0c -> ../../sda
 |    |-- ata-VBOX_HARDDISK_VB946693c0-53b88d0c-part1 -> ../../sda1
 |    |-- ata-VBOX_HARDDISK_VB946693c0-53b88d0c-part2 -> ../../sda2
 |    |-- ata-VBOX_HARDDISK_VB946693c0-53b88d0c-part5 -> ../../sda5
 |-- by-path
 |    |-- pci-0000:00:01.1-scsi-1:0:0:0 -> ../../sr0
 |    |-- pci-0000:00:0d.0-scsi-0:0:0:0 -> ../../sda
 |    |-- pci-0000:00:0d.0-scsi-0:0:0:0-part1 -> ../../sda1
 |    |-- pci-0000:00:0d.0-scsi-0:0:0:0-part2 -> ../../sda2
 |    `-- pci-0000:00:0d.0-scsi-0:0:0:0-part5 -> ../../sda5
 `-- by-uuid
      |-- abd5f613-c67e-4ef6-ba44-bdffeee3a8aa -> ../../sda5
      `-- df7df68b-6c55-48d8-96bb-090280834f68 -> ../../sda1
```

```
ENV{ID_FS_USAGE}=="filesystem|other|crypto", ENV{ID_FS_UUID_ENC}=="?*",↩
      SYMLINK+="disk/by-uuid/$env{ID_FS_UUID_ENC}"
```

```
ENV{ID_FS_USAGE}=="filesystem|other", ENV{ID_FS_LABEL_ENC}=="?*", ↩
     SYMLINK+="disk/by-label/$env{ID_FS_LABEL_ENC}
```

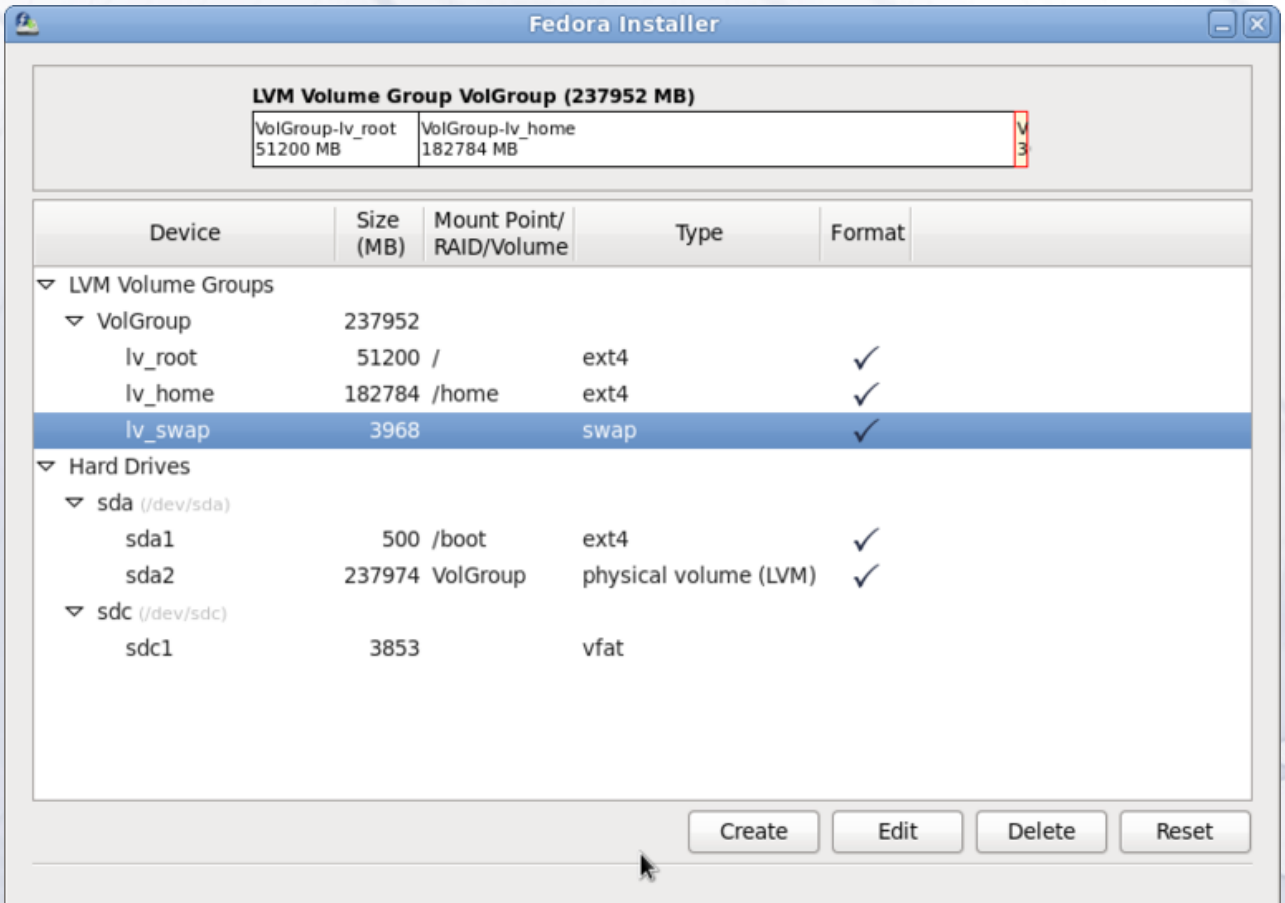
El montaje del sistema de archivos

Esquema general del arranque de Linux



Los sistemas de archivos se montan

```
$ mount
/dev/sda1 on /          type ext4      (rw,errors=remount-ro,commit=0)
proc      on /proc      type proc      (rw,noexec,nosuid,nodev)
none      on /sys       type sysfs     (rw,noexec,nosuid,nodev)
none      on /dev       type devtmpfs  (rw,mode=0755)
none      on /var/run   type tmpfs     (rw,nosuid,mode=0755)
none      on /var/lock  type tmpfs     (rw,noexec,nosuid,nodev)
```



El kernel siempre necesita que se le indique un root que sea capaz de leer.

- El root se indica mediante mediante la variable root.
- El kernel sólo entiende los nombres standard *root=/dev/sda1*, *root=/dev/sdb2*.
- Podemos cargar un initial ram disk. Un fichero con un sistema de archivos comprimido que temporalmente se alojará en la ram.

initramfs

El kernel 2.6 siempre lleva un initramfs incrustado (vacio).

Un CPIO gzipeado en el caso del initramdisk o la imagen de una partición legible como ext3 en el caso de un initrd

```
$ gunzip -c /boot/initramfs-2.6.32-5-686.img | cpio -i -d -H newc --no-↩
    absolute-filenames
72042 blocks
```

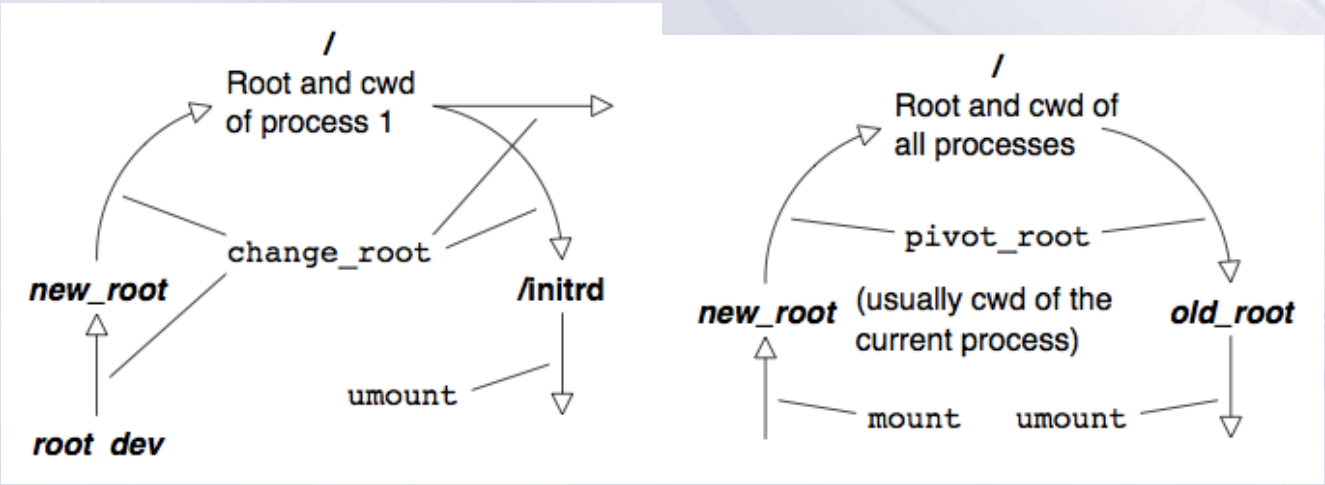
Para construir el initramfs tenemos generadores como mkinitramfs (Debian) o Dracut (sponsorizado por Red Hat).

- Crean un sistema mínimo instalando todo lo requerido: aplicaciones básica, bibliotecas, módulos.
- Contienen la funcionalidad requerida para montar el root.

Mecanismos de pivotaje

En todos los casos el problema es asegurarnos de que no nos hemos dejado ningún proceso con algún descriptor de fichero abierto.

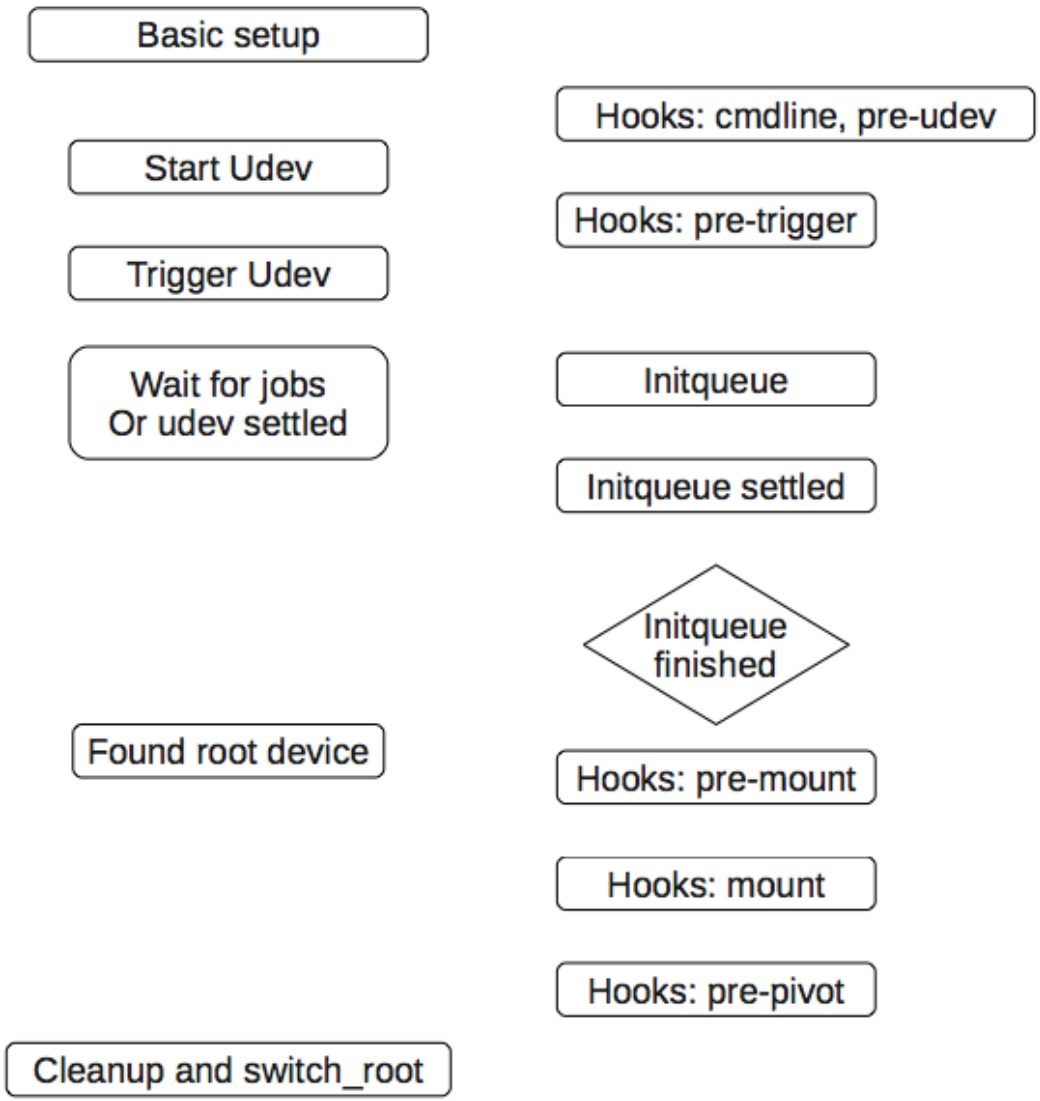
- change_root. Descrito como intentar cambiar la alfombra debajo de tus mientras mientras saltas.
  Si nos queda algún proceso el mecanismo falla.
- pivot_root. Intercambia dos roots.
  Si queda algún proceso no podemos desmontar el volumen y liberar la memoria.
- switch_root. El más moderno. Simplemente elimina todos los contenidos del initial ramdisk (tmpfs) liberando su memoria y ejecuta un chroot.
  Si queda algún proceso no podemos borrar y liberar ese fichero.



Dracut. Udev based initial ramdisk generator

- Event based with udev
  - udev, an event-driven hotplug agent, which invokes helper programs as hardware devices, disk partitions and storage volumes matching certain rules come online.
  - This allows discovery to run in parallel, and to progressively cascade into arbitrary nestings of LVM, RAID or encryption to get at the root file system.
- Generic across distributions and hardware
  - Boot any system configuration on any hardware with the same initramfs image
- Uses the tools installed on the system
- Nothing hardcoded
- Easy to extend and customize

Esquema de la ejecución de Dracut



```
# Arrancamos udev y ordenamos generar los eventos que inicializarán los
# dispositivos
udevd --daemon
udevadm control --reload
udevadm trigger --type=subsystems --action=add
udevadm trigger --type=devices    --action=add

while true; do

    # Comprobamos si se satisface la condición de terminación
    for f in $hookdir/initqueue/finished/*.sh; do
        . "$f" || return 1
    done && break

    # Ordenamos procesar la cola de eventos pendientes y esperamos
    udevadm settle --exit-if-exists=$hookdir/initqueue/work
                                    $settle_exit_if_exists

    # Procesamos los scripts ayudantes que estaban instalados
    # o que acaban de ser instalados por alguna regla.
    for job in $hookdir/initqueue/*.sh; do
        . $job
    done

    for job in $hookdir/initqueue/settled/*.sh; do
        . $job
    done
done
```

Es casi pseudocódigo, está simplificado para que se entienda la idea y me entre en la transpa 0:-)

Ejemplo de un script pre-trigger

Añadimos una regla que crea un enlace del volumen que buscamos a /dev/root:

```
cat <<EOF >> /etc/udev/rules.d/99-root.rules

KERNEL=="${root#block:/dev/}", SYMLINK+="root"
SYMLINK=="${root#block:/dev/}", SYMLINK+="root"

EOF
```

Instalamos 2 scripts en la initqueue que comprueban que el volumen /root se ha montado.

```
cat <<EOF > $hookdir/initqueue/settled/blocksymlink.sh

if [ -e "${root#block:}" ]; then
    ln -s "${root#block:}" /dev/root;
    rm "$job"
fi
EOF

cat <<EOF >> "$hookdir/initqueue/finished/devexists-${_name}.sh"

[ -e "${root#block:}" ]

EOF
```

Bibliografía

Papers from Ottawa Linux Symposium

- Booting Linux: The History and the Future
  Werner Almesberger, 2000
- kboot — A Boot Loader Based on Kexec
  Werner Almesberger, 2006
- Kdump: Smarter, Easier, Trustier
  Vivek Goyal, Neil Horman, Kenichi Ohmichi, Maneesh Soni, Ankita Garg, 2007
- udev – A Userspace Implementation of devfs
  Greg Kroah-Hartman, 2003
- Kernel Support for Stackable File Systems
  Josef Sipek, Yiannis Pericleous, and Erez Zadok, 2007

- Master boot record
  http://en.wikipedia.org/wiki/Master_boot_record
- int 13h
  http://en.wikipedia.org/wiki/INT_13h
- int 10h
  http://en.wikipedia.org/wiki/INT_10h
- GUID Partition Table
  http://en.wikipedia.org/wiki/GUID_Partition_Table
- Unified Extensible Firmware Interface
  http://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface

Grub

- Booting Linux on x86 using Grub2
  http://people.apache.org/~skitching/MineOfInformation/linux/Booting_Linux_on_x86_with_Grub2.html

- Details of GRUB on the PC
  http://www.pixelbeat.org/docs/disk/

- GRUB GPT HOWTO
  http://www.wensley.org.uk/gpt

- Grub Manual
  http://www.gnu.org/software/grub/manual/html_node/index.html

- Booting Linux natively on a UEFI system (without BIOS CSM) using GRUB2
  https://help.ubuntu.com/community/UEFIBooting

Dracut

- ramfs filesystem
  http://www.kernel.org/doc/Documentation/filesystems/ramfs-rootfs-initramfs.txt
- tmpfs filesystem
  http://www.kernel.org/doc/Documentation/filesystems/tmpfs.txt
- Dracut - A Generic Initramfs Infrastructure
  Harald Hoyer. Talk at FOSDEM 2010
- modules.d/99base/init.sh
  http://git.kernel.org/?p=boot/dracut/dracut.git;a=blob;f=modules.d/99base/init.sh;h=88ec184616f8d7a7c29a1d29a62c8830aad4c884;hb=HEAD
- http://fedoraproject.org/wiki/Dracut
- Dracut manual
  http://www.kernel.org/pub/linux/utils/boot/dracut/dracut.html

/var and /usr move

- Introducing /run
  http://lwn.net/Articles/436012/
- Fedora 17 - Features - Move all to /usr
  http://fedoraproject.org/wiki/Features/UsrMove
- Systemd - Booting Without /usr is Broken
  http://www.freedesktop.org/wiki/Software/systemd/separate-usr-is-broken
- Debian Read Only Root
  https://wiki.debian.org/ReadonlyRoot
- debian-devel mailing list - Move all to /usr
  http://thread.gmane.org/gmane.linux.debian.devel.general/165785