

# Arranque de un Sistema Linux



Julio Sánchez Cubas  
Senior IT Architect. IBM España  
Member, SPIGIT Technical Expert Council (TEC)  
E-mail: [jscubas@es.ibm.com](mailto:jscubas@es.ibm.com)

# Agenda

- Introducción
  - ¿Qué es el Boot?
  - Plataformas Linux
- Fases del Arranque en Linux
  - FASE 1: Hardware
  - FASE 2: BootLoader
  - FASE 3: Kernel
  - FASE 4: Init
- Apéndices
- Ruegos y Preguntas



# Introducción

## Objetivos:

- Comprender qué sucede en nuestro sistema desde que pulsamos el botón de encendido hasta que el sistema operativo está completamente cargado.
- Entender las distintas fases que atraviesa el sistema y los distintos ficheros y comandos que están involucrados.



# Una muy breve historia de GNU/Linux

1985: GNU (Richard Stallman).Emacs, gcc

1989: Minix (Andrew S. Tanenbaum)

1991: Linux 0.1 (Linus Torvalds)  
Crecimiento MS-DOS  
MacOS  
UNIX

<http://www.kernel.org>

1992-3: Inclusión de componentes clave  
(TCP/IP, X-Windows.

1993: Primera distribución Linux  
(Slackware)

<http://www.slackware.com>

1994: Versión 1.0

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)  
Newsgroups: comp.os.minix  
Subject: What would you like to see most in minix?  
Summary: small poll for my new operating system  
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>  
Date: 25 Aug 91 20:57:08 GMT  
Organization: University of Helsinki

Hello everybody out there using minix -I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things). I've currently ported bash(1.08) and gcc(1.40),and things seem to work.This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)  
Linus (torvalds@kruuna.helsinki.fi)  
PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-).



# Arquitecturas que soportan Linux

- Linux actualmente corre sobre:
  - [Acorn](#): [Archimedes](#), [A5000](#) y las series [RiscPC](#): ([ARM](#), [StrongARM](#), [Intel XScale](#) etc.)
  - [AMD64](#): Procesadores de [AMD](#) con tecnología de 64-bits (conocidos inicialmente como x86-64)
  - [Axis Communications](#): [CRIS](#)
  - [Compaq](#): [Alpha](#)
  - [Hewlett Packard](#): familia [PA-RISC](#)
  - [Hitachi](#): [SuperH](#) ([SEGA Dreamcast](#)), [H8/300](#)
  - [IA-64](#): PCs con tecnología de 64-bits [Intel Itanium](#)
  - [zSeries](#): [IBM zSeries](#) (z800, z890, z900, z990) y virtualizado bajo el sistema operativo z/VM.
  - [Intel](#): 80386 y superiores: [IBM PCs](#) y compatibles: [80386](#), [80486](#), la serie [Pentium](#) completa; [AMD Athlon](#), [Duron](#), [Thunderbird](#); las series [Cyrix](#). El soporte para microprocesadores Intel [8086](#), [8088](#), [80186](#), [80188](#) e [80286](#) está siendo desarrollado (véase el proyecto [ELKS](#))
  - [Microsoft](#): [Xbox](#)
  - [MIPS](#): estaciones [Silicon Graphics, Inc.](#), ...
  - [Motorola](#): [68020](#) y superiores: modelos nuevos de [Amiga](#)
  - [Apple](#): algunas computadoras
  - [NEC Corporation](#): [v850e](#)
  - [PowerPC](#) y [POWER](#): la mayoría de las nuevas [Apple](#) (todas las basadas en PCI [Power Macintosh](#), soporte limitado para las viejas [NuBus Power Macs](#)), clones de Power Mac vendidos por [Power Computing](#), [UMAX](#) y [Motorola](#), [Amigas](#) mejorados con placas "Power-UP" (como [Blizzard](#) o [CyberStorm](#)), [IBM RS/6000](#), sistemas [iSeries](#) y [pSeries](#), numerosas plataformas PowerPC embebidas
  - [Sony](#): [PlayStation 2](#)
  - [SPARC](#) y [UltraSparc](#): puestos de trabajo [Sun](#), y sus clones hechos por [Tatung](#) y otros



## ¿Qué es el Boot?

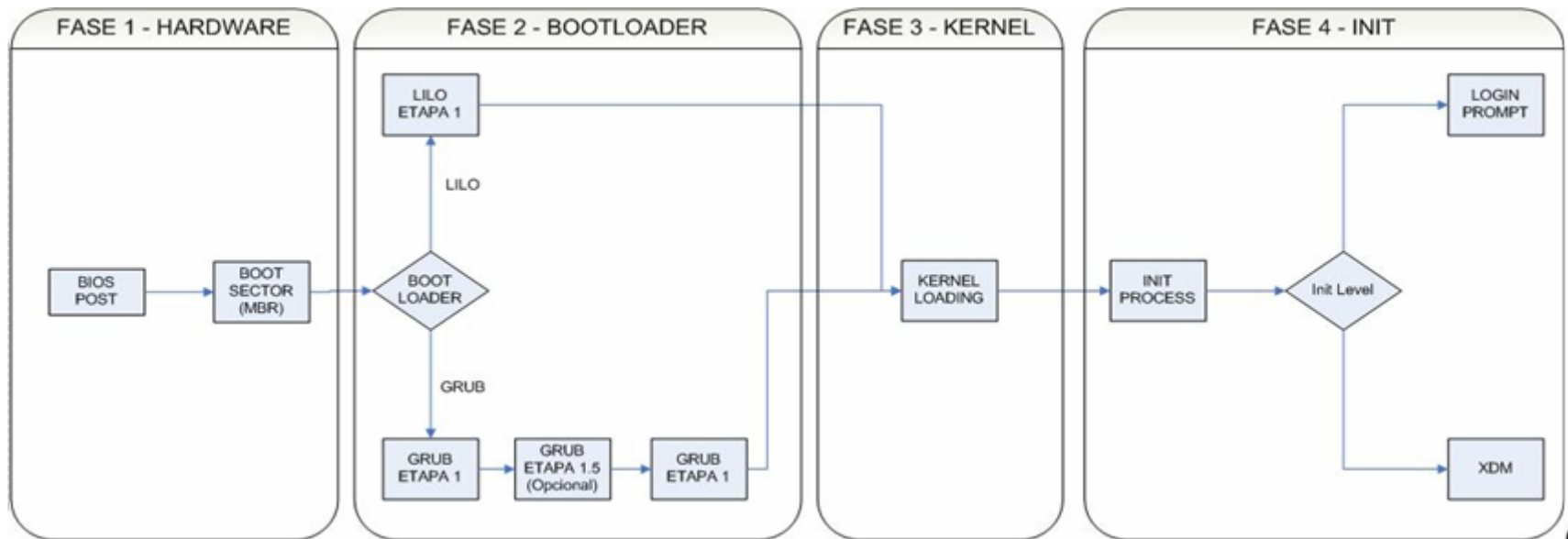
- El término Bootstrapping alude a una leyenda alemana sobre el Barón de Münchhausen, el cual era capaz de salir de una ciénaga tirándose asimismo de su propio pelo. En versiones posteriores, el utilizó los cordones de sus zapatos (boot straps) para salir del mar, lo que dio lugar al término bootstrapping.
- En informática, el término se utilizar para cualquier proceso en el que un sistema simple activa a otro más complejo. Es el concepto de arrancar un sistema a partir de una parte pequeña del mismo.
- Algunos ejemplos:
  - En informática general se utiliza como sinónimo de *boot* o *booting*, para expresar el proceso de arranque de un ordenador. (GRUB, LILO, NTLDR, etc.)
  - Se utiliza en compiladores para definir aquellos compiladores que están desarrollados con el mismo lenguaje que compilan (gcc, Ocaml, PL/I)

<http://en.wikipedia.org/wiki/Bootstrap>



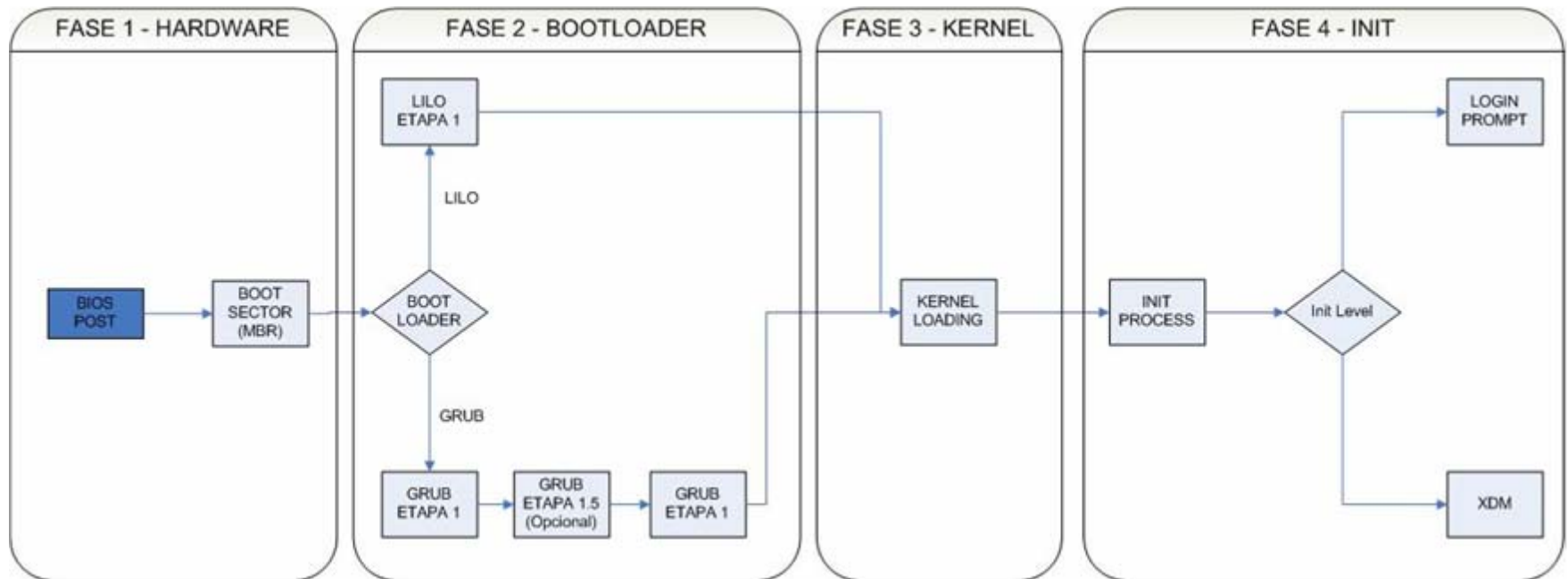
# Fases del Arranque en Linux

- FASE 1: Hardware
- FASE 2: BootLoader
- FASE 3: Kernel
- FASE 4: Init



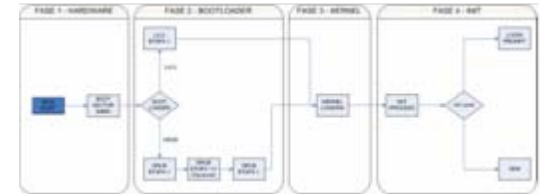


# FASE 1: Hardware





# FASE 1: Hardware



- En esta fase el sistema se inicia pasando el control a la BIOS
- BIOS: Basic Input/Output System

La BIOS es un pequeño programa que se encuentra grabado en una memoria en la placa base.

Esta memoria ha evolucionado desde ROM->PROM->EPROM->Memoria Flash.

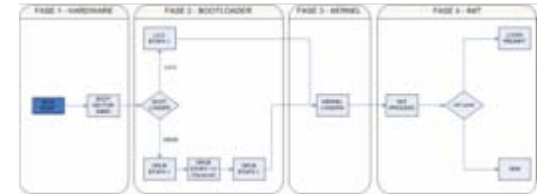
Alimentado continuamente (batería)

Guarda la configuración de nuestro sistema.

Realiza POST



# Formatos de BIOS más comunes



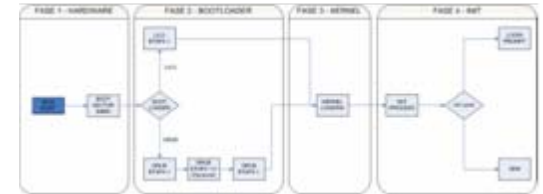
- Tradicional IBM System Bios
- AMI Bios
- Phoenix Bios
- Linux Bios
  
- (Ver apéndice)

```

Full Screen LOGO Show      [Disabled]
▶ Hard Disk Boot Priority  [Press Enter]
Virus Warning              [Disabled]
CPU Internal Cache        [Enabled]
External Cache            [Enabled]
Quick Boot                [Enabled]
1st Boot Device           [Hard Disk]
2nd Boot Device           [Hard Disk]
3rd Boot Device           [CDROM]
Boot Other Device         [Enabled]
Seek Floppy               [Disabled]
Boot Up Num-Lock LED      [On]
Gate A20 Option           [Fast]
Typematic Rate Setting    [Disabled]
x Typematic Rate (Chars/Sec) 6
x Typematic Delay (Msec) 250
APIC Function              [Enabled]
MPS Table Version         [1.4]
Boot OS/2 for DRAM > 64MB [No]
    
```



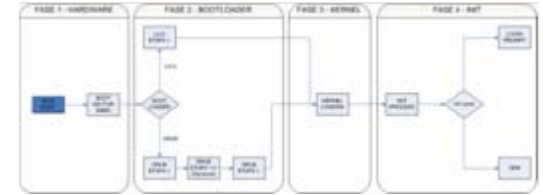
# Linux Bios



- LinuxBios es un proyecto Open Source que tiene como objetivo sustituir la BIOS normal con una Bios con una pequeña inicialización de Hardware y un kernel de Linux comprimido.
- El proyecto comenzó como parte de un trabajo de clustering en el laboratorio Nacional de Los Álamos. El objetivo era conseguir que cuando se arranque un nuevo nodo pudiera añadirse al cluster.
- Beneficios fundamentales:
  - Evita la duplicidad de proceso (BIOS y kernel)
  - Minimizar el tiempo de arranque
  - Liberarse de código propietario y desconocido en laBIOS
  - Evitar la necesidad de BootLoaders.



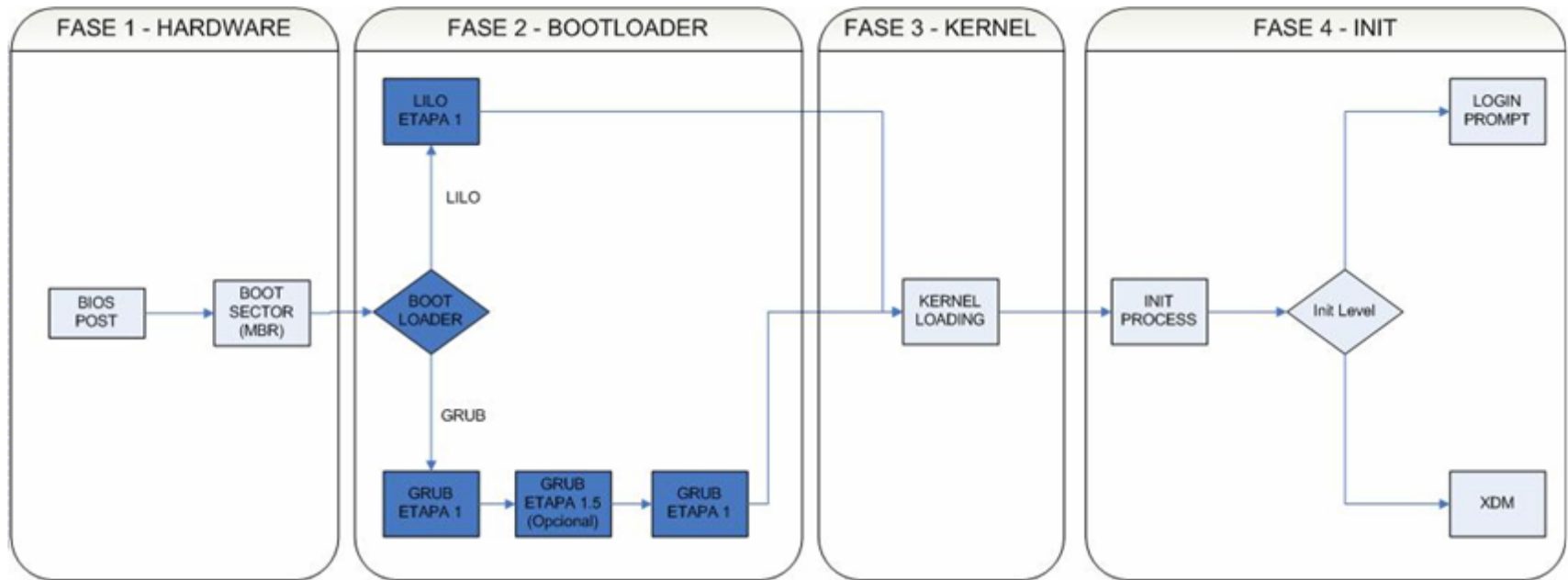
# POST



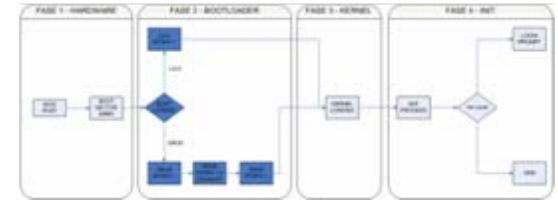
- POST: Power On Self Test, también llamado IPL en otras arquitecturas (No PC).
- Tareas que realiza:
  1. Verifica la integridad del código de la BIOS
  2. Determina porqué se ejecuta el POST (arranque en frío, Soft reset, error, stand-by, hibernación, etc.)
  3. Busca, dimensiona y verifica la memoria del sistema (RAM y ROM)
  4. Busca, inicializa y cataloga los buses y dispositivos del sistema
  5. (Opcional) Pasa el control a otras BIOS especializadas de dispositivos (Red, RAID, etc.)
  6. Proporciona la interfaz de usuario para configurar parámetros del sistema (Velocidad de CPU, orden de arranque, “tunning” del sistema, etc.)
  7. Identifica, organiza y selecciona los dispositivos de arranque disponibles
  8. Comienza el proceso de arranque del sistema, llamando al bootloader



# FASE 2: BootLoader



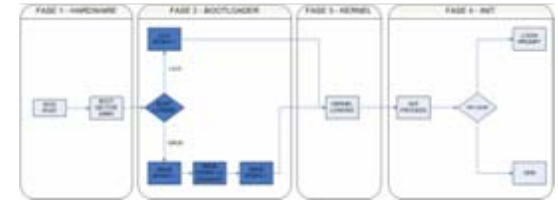
## FASE 2: BootLoader



- El objetivo del Bootloader es cargar parte del nucleo (kernel) del sistema operativo en memoria y ejecutarlo.
- A partir de cargarse, el Bootloader toma el control y se encarga de cargar el resto del sistema operativo



# Bootloader – Ubicaciones (1)



- En un **disquete**:

El primer sector de un disquete está reservado como sector de arranque (boot)

- En el **disco duro**:

Puede ubicarse en el primer sector de cada una de las particiones del disco

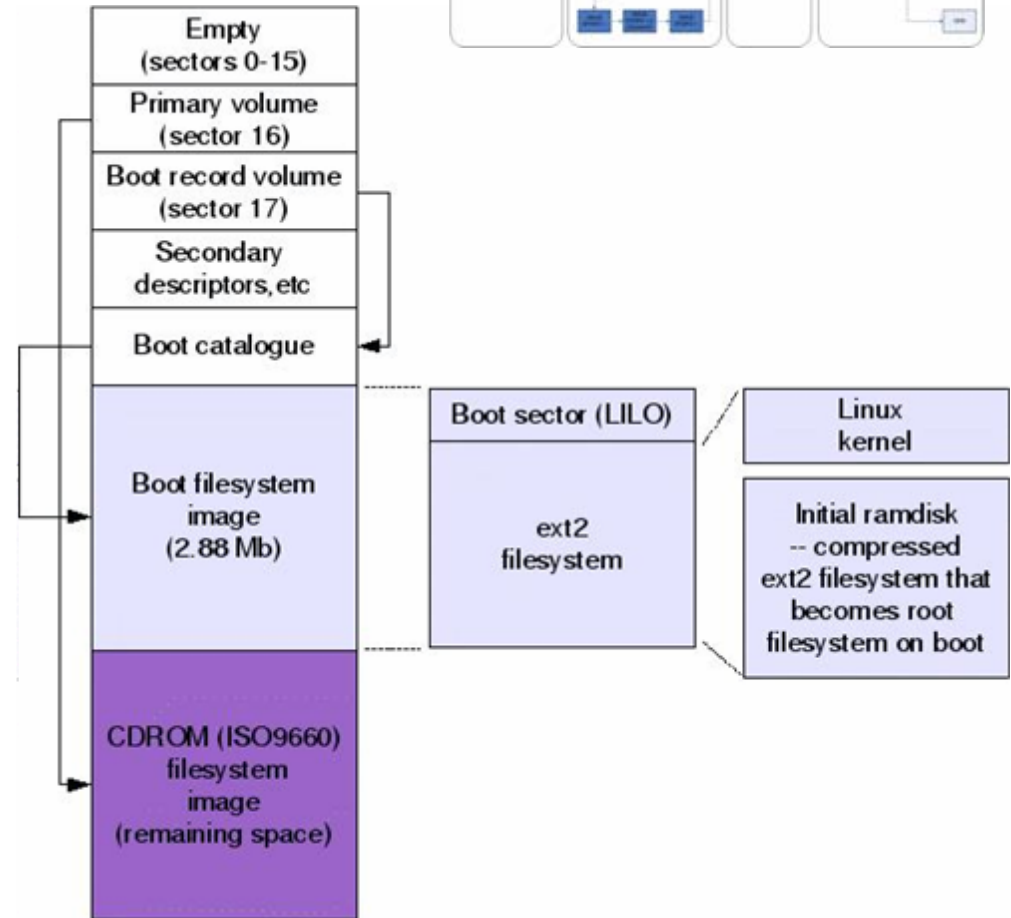
A parte el primer sector del disco existe un sector de arranque global (Master Boot Record o MBR). Este es el lugar más común para instalarlo.



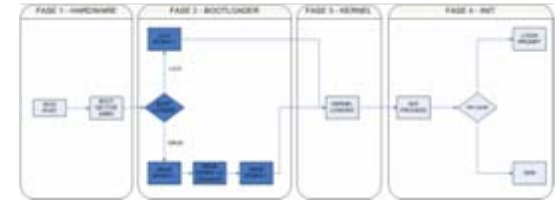


## Bootloader – Ubicaciones (2)

- En un **CD-ROM**
  - Siguiendo la especificación “El Torito”
  - Es parte del proyecto SYSLINUX
  - Utiliza una configuración especial en el CDROM.
- BootLoader cargado **desde la red**
  - Posible con LinuxBios
  - En otras plataformas no PC más común
  - Algunas tarjetas de red están preparadas
  - Normalmente via broadcasting (RARP)



# El proyecto SYSLINUX

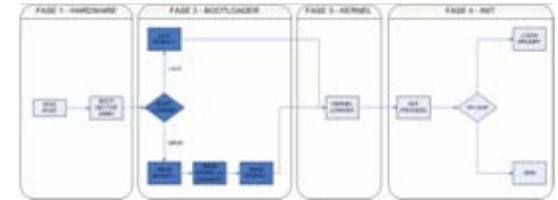


Este proyecto cubre bootloaders ligeros para:

- Floppy disks: SYSLINUX
- Arranque en red: PXELINUX
- Arranque desde CD-ROM (“El Torito”): ISOLINUX
- Arranque de sistemas Legacy (como dos) via PXE.



# BootLoaders en Linux



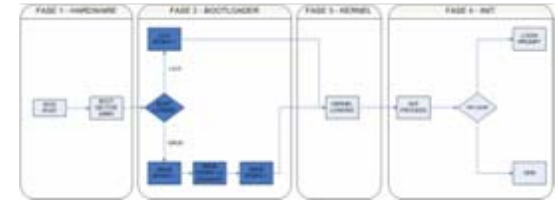
- LILO. The **L**inux **L**oader
- GRUB. **G**Rand **U**nifying **B**ootloader

Ambos son capaces de cargar tanto sistemas Linux como otros sistemas operativos

Ambos se suelen ubicar en el sector de arranque de disquetes o en el MBR del disco duro



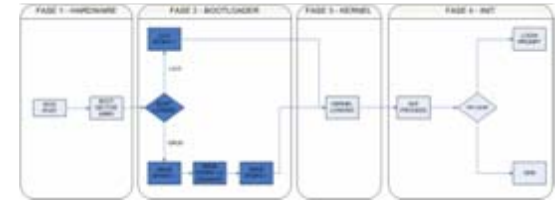
# LILO



- Bastante rudimentario
- Bootloader de una sola etapa.
- No entiende de sistemas operativos, ni de sistemas de ficheros
- Lee datos del disco utilizando llamadas nativas de la BIOS, con punteros directos a los ficheros que necesita
- Los almacena a través de un fichero mapa que se almacena en el sector de arranque



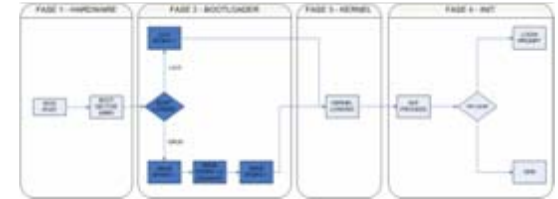
# Fases de LILO



1. El firmware carga el sector de arranque de LILO y lo ejecuta
2. LILO carga su fichero de mapa por medio de llamadas de la BIOS, este muestra el prompt de opciones a cargar
3. El usuario selecciona el kernel que arrancar
4. LILO carga el kernel seleccionado por medio de llamadas de la BIOS y utilizando los parámetros de ubicación en el fichero de mapa
5. (Opcional) LILO carga un ramdisk
6. LILO ejecuta el kernel indicando donde está el root fs y si es necesario el ramdisk



# Ficheros de LILO



- Ejemplo de /etc/lilo.conf

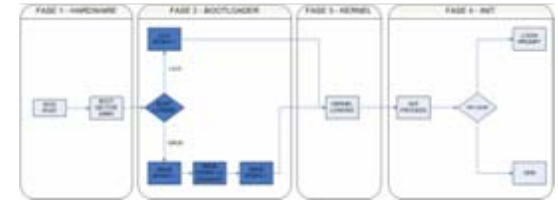
```
boot=/dev/hda2
root=/dev/hda2
install=/boot/boot.b
map=/boot/map
vga=normal
delay=20
image=/vmlinuz
    label=Linux
    read-only
other=/dev/hda1
    table=/dev/hda
    label=win
```

- Para cargar la configuración hay que ejecutar el comando lilo.

```
# lilo /etc/lilo.conf
```



# GRUB

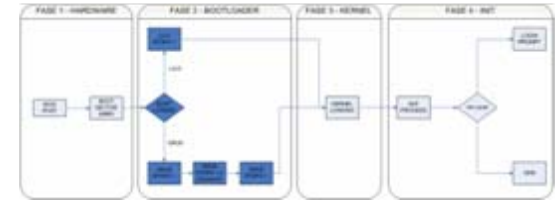


- Más avanzado (y más moderno) que LILO
- Trabaja en dos o tres etapas (Stages)
- Tiene capacidad para arrancar un kernel via red
- En cada etapa va cargando más elementos para arrancar
- Entiende de sistemas de ficheros
- Permite especificar parámetros de forma dinámica en el arranque, no utiliza valores estáticos





# Fases de GRUB

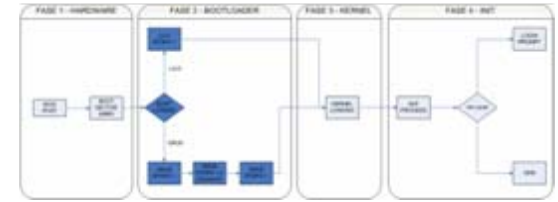


1. Etapa 1: El firmware carga el sector de arranque de GRUB en memoria. (En él están los números de bloque del disco donde se encuentra la siguiente etapa)
2. Etapa 1.5: (Se denomina así porque esta etapa podría ser opcional). Su objetivo es cargar el código que reconoce sistemas de ficheros y a partir de ahí cargar la etapa 2 como un fichero.
3. Etapa 2: GRUB muestra el menú con las opciones de boot que hayamos definido y un prompt donde podemos especificar ramdisks, kernels, etc. a cargar.
4. GRUB ejecuta los comandos introducidos, bien las definidas por nosotros en el fichero de configuración (`grub.conf`, `menu.lst`) y comienza la carga del kernel.

NOTA: Aquí reside la potencia de GRUB, es capaz de cargar ficheros realizar tareas dinámicas en la fase de arranque del sistema



# Ficheros de GRUB

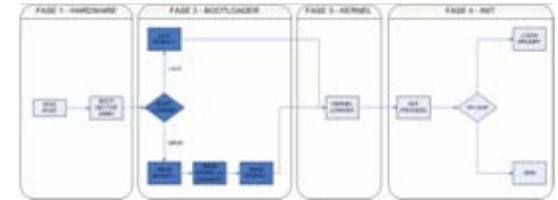


- Depende de la distribución pero suele ser similar a esto:

```
ninakula:/boot/grub # ls -la
total 764
drwxr-xr-x  2 root root   4096 Feb 17  2005 .
drwxr-xr-x  3 root root   4096 Feb 17  2005 ..
-rw-r--r--  1 root root     60 Feb 17  2005 device.map
-rw-r--r--  1 root root   8244 Jun 30  2004 e2fs_stage1_5
-rw-r--r--  1 root root   8036 Jun 30  2004 fat_stage1_5
-rw-r--r--  1 root root   7284 Jun 30  2004 ffs_stage1_5
-rw-r--r--  1 root root   8768 Jun 30  2004 jfs_stage1_5
-rw-----  1 root root    787 Feb 17  2005 menu.lst
-rw-r--r--  1 root root   7476 Jun 30  2004 minix_stage1_5
-rw-r--r--  1 root root 184292 Jun 30  2004 nbgrub
-rw-r--r--  1 root root 185316 Jun 30  2004 pxegrub
-rw-r--r--  1 root root   9716 Jun 30  2004 reiserfs_stage1_5
-rw-r--r--  1 root root    512 Jun 30  2004 stage1
-rw-r--r--  1 root root 101650 Feb 17  2005 stage2
-rw-r--r--  1 root root 184420 Jun 30  2004 stage2.netboot
-rw-r--r--  1 root root   6932 Jun 30  2004 vstafs_stage1_5
-rw-r--r--  1 root root   9596 Jun 30  2004 xfs_stage1_5
ninakula:/boot/grub #
```



## Un ejemplo de configuración de GRUB



- El fichero menu.lst (o grub.conf)
- GRUB es interactivo, para configurar hay que ejecutar grub y desde su prompt utilizar los comandos.

```

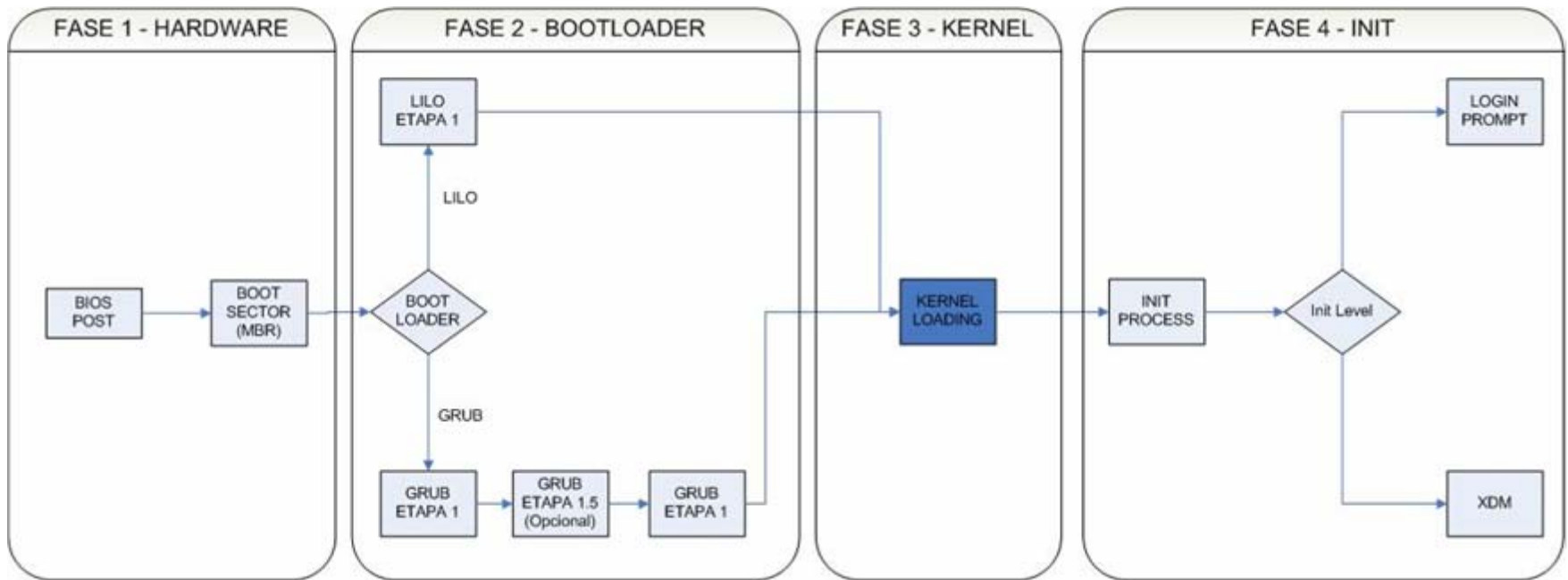
color white/blue black/light-gray
default 0
timeout 8
gfxmenu (hd0,0)/boot/message
█
title Linux
    kernel (hd0,0)/boot/vmlinuz root=/dev/sda1 vga=0x317 selinux=0 splash=silent
    resume=/dev/sda2 elevator=cfq showopts
    initrd (hd0,0)/boot/initrd

title Disquete
    root (fd0)
    chainloader +1

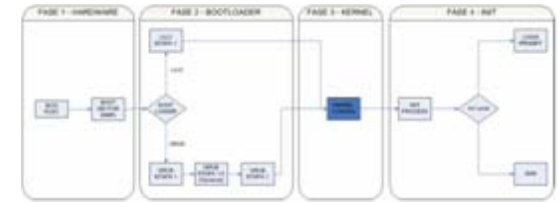
title Failsafe
    kernel (hd0,0)/boot/vmlinuz root=/dev/sda1 showopts ide=nodma apm=off acpi=off
    vga=normal noresume selinux=0 barrier=off nosmp noapic maxcpus=0 3
    initrd (hd0,0)/boot/initrd
  
```



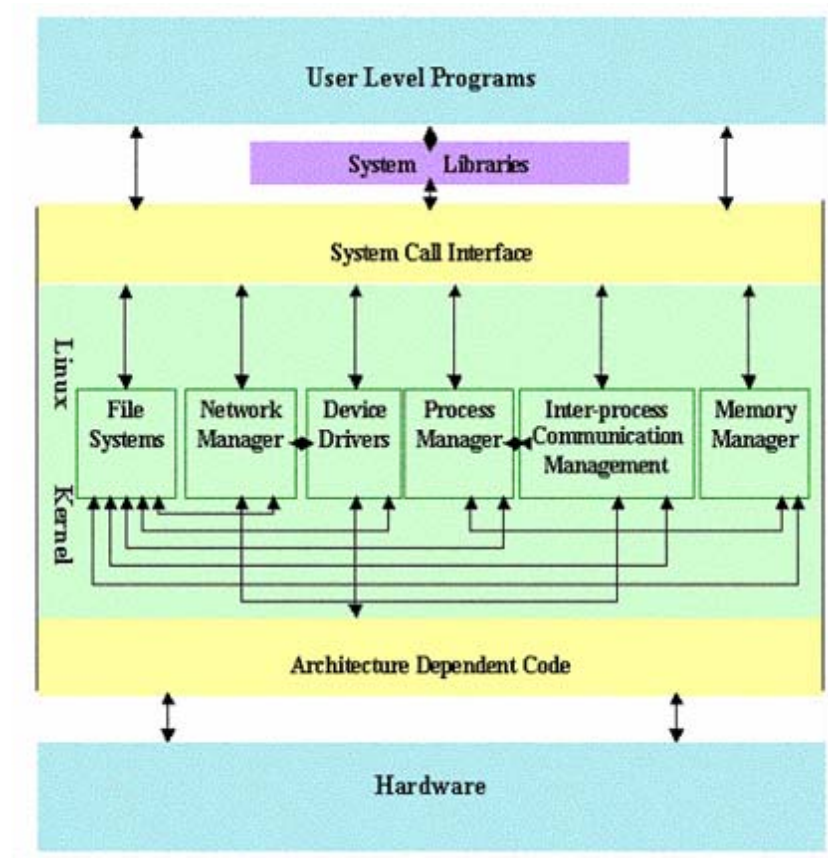
# FASE 3: Kernel



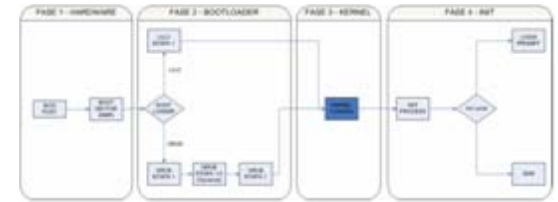
## Breve descripción de la Arquitectura del Kernel Linux



- Arquitectura Monolítica
- Es un largo y complejo programa compuesto de un gran número de subsistemas lógicos
- Gestionado directamente por Linus Torvalds
- Con capacidad de carga de Módulos
- Si bien por definición está formado por una capa lógica, internamente funciona con más.



## Kernel Monolítico vs MicroKernel



- La mayor parte de los kernels Unix son monolíticos
  - > Cada capa del kernel está integrada dentro del propio programa del kernel y se ejecuta en Modo Kernel con independencia del proceso que la utilice
- Las arquitecturas Microkernel (como Mach de GNU o Minix) solicitan un pequeño conjunto de funciones al kernel y sus componentes para implementar la mayor parte de las funciones.
- Los sistemas operativos Microkernel obligan a mantener interfaces muy bien definidos y estables entre sus componentes individuales así como el poder realizar optimizaciones sofisticadas, lo cual suele redundar en rendimientos más bajos que los kernels monolíticos.

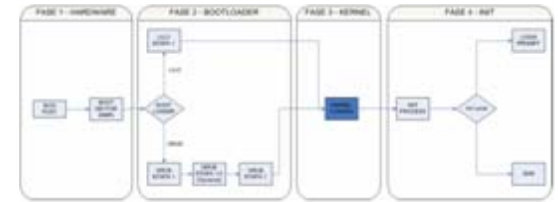
*" I still maintain the point that designing a monolithic kernel in 1991 is a fundamental error. Be thankful you are not my student. You would not get a high grade for such a design :-)"*

*(Andrew Tanenbaum to Linus Torvalds)*





## FASE 3: Kernel

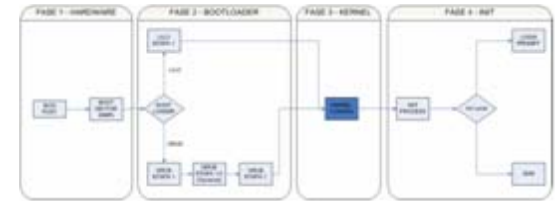


- En esta fase comienza la ejecución del kernel, descomprimiéndose a sí mismo. (Esto es código en el principio de la propia imagen del kernel)
- Comienza la inicialización del kernel y el chequeo y puesta en marcha de algunos de los dispositivos para los que se ha dado soporte.
  - Detecta la CPU y su velocidad
  - Inicializa el Display para mostrar información por pantalla
  - Comprueba el bus PCI e identifica y crea una tabla con los periféricos conectados (muestra por pantalla los mismos y su estado)
  - Inicializa el sistema de gestión de memoria virtual, incluyendo el swapper
  - Inicializa todos los periféricos compilados dentro del kernel (no como módulos), normalmente sólo se configuran así los periféricos necesarios para esta fase del arranque, el resto se configuran como módulos
  - Monta el sistema de ficheros root (“/”)
  - A partir de aquí llama al proceso init que se ejecuta con uid 0 y será el padre de todos los demás procesos





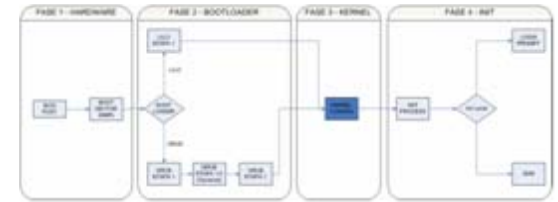
# Configuración del Kernel



- Debemos instalar los fuentes del kernel. Bien con el tar.gz, paquete deb o rpm.
- Ir al Directorio `/usr/src/linux`
- Configurar los elementos del kernel
  - `make config` ó
  - `make menuconfig` ó
  - `make xconfig`
- Realizar las modificaciones de elementos a soportar, módulos a cargar, etc.
- Recompilar kernel y módulos
  - `make clean dep bzImage`
  - `make modules`
- Instalar el kernel y los módulos
  - `make install` y/ó
  - `make modules_install`



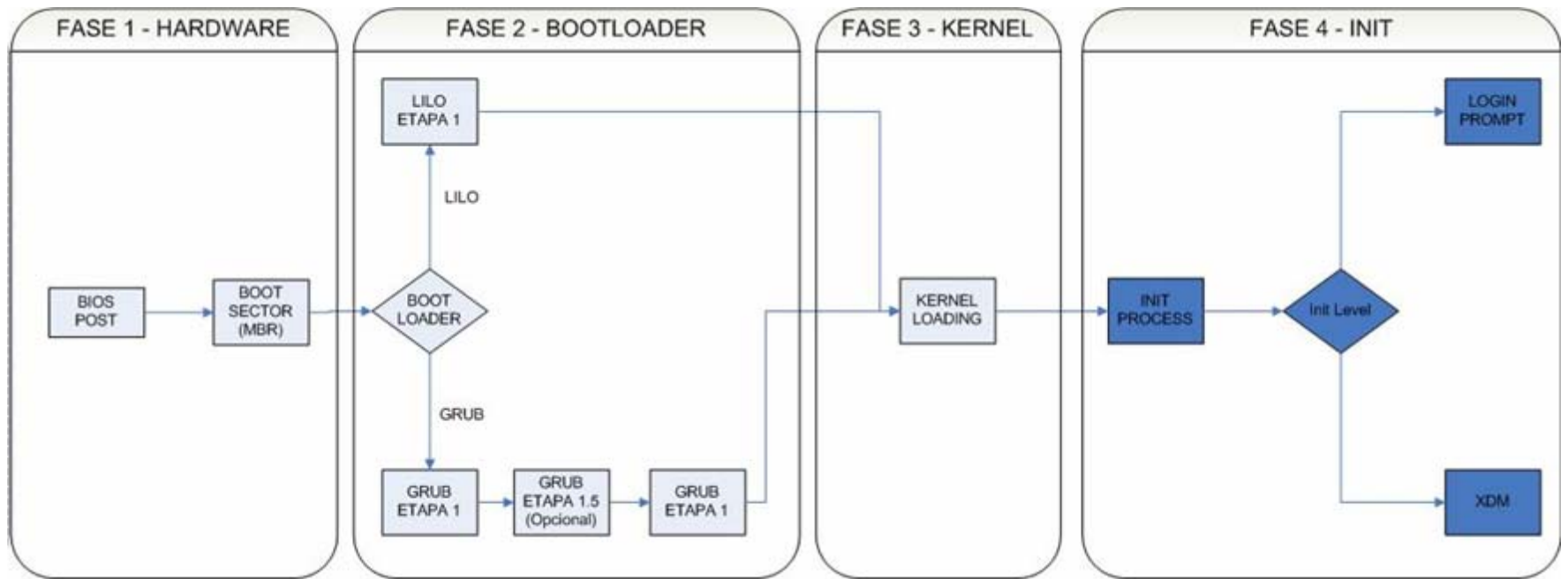
# Ramdisks



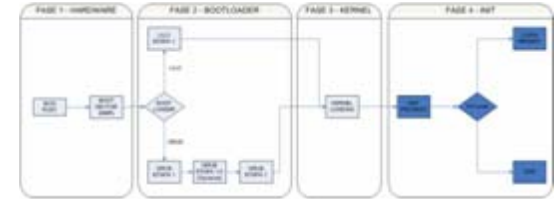
- Son discos virtuales creados utilizando la memoria RAM del sistema
- Como consecuencia son discos muy rápidos y de tamaño normalmente pequeño
- Son volátiles, es decir sus datos no persisten tras un apagado o reinicio
- Sirven para varias utilidades:
  - Como cachés en Servidores Web
  - Como cualquier disco con datos de alta velocidad
  - En nuestro caso concreto se utilizan para descomprimir el kernel de forma rápida y cargar los módulos en un sistema montado.



# FASE 4: Init



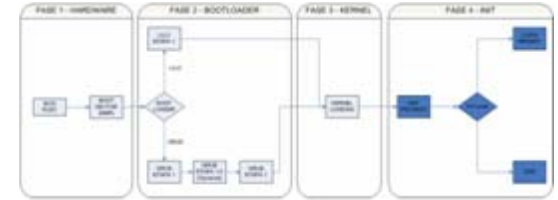
## FASE 4: Init



- En estos momentos el kernel está cargado, tenemos gestión de memoria, una parte del hardware está inicializado y tenemos un sistema de ficheros root.
- A partir de ahora el resto de operaciones se van a realizar directa o indirectamente por el proceso init.
- El proceso init lee del fichero `/etc/inittab` la configuración a utilizar.
- Ejecuta el comando `/etc/rc.sysinit` el cual realiza una inicialización básica del sistema
- En función del runlevel ejecuta los comandos establecidos



# Ejemplo de inittab (1)



```
# The default runlevel is defined here
id:5:initdefault:
```

```
# First script to be executed, if not booting in emergency (-b)
mode
si::bootwait:/etc/init.d/boot
```

```
# /etc/init.d/rc takes care of runlevel handling
```

```
#
```

```
# runlevel 0 is System halt (Do not use this for initdefault!)
```

```
# runlevel 1 is Single user mode
```

```
# runlevel 2 is Local multiuser without remote network (e.g. NFS)
```

```
# runlevel 3 is Full multiuser with network
```

```
# runlevel 4 is Not used
```

```
# runlevel 5 is Full multiuser with network and xdm
```

```
# runlevel 6 is System reboot (Do not use this for initdefault!)
```

```
#
```

```
l0:0:wait:/etc/init.d/rc 0
```

```
l1:1:wait:/etc/init.d/rc 1
```

```
l2:2:wait:/etc/init.d/rc 2
```

```
l3:3:wait:/etc/init.d/rc 3
```

```
#l4:4:wait:/etc/init.d/rc 4
```

```
l5:5:wait:/etc/init.d/rc 5
```

```
l6:6:wait:/etc/init.d/rc 6
```

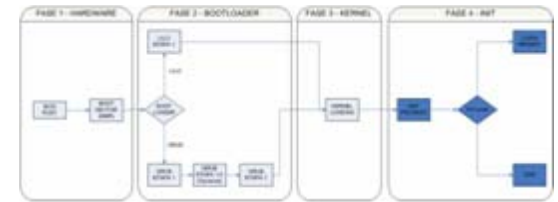
```
# what to do in single-user mode
```

```
ls:S:wait:/etc/init.d/rc S
```

```
~~:S:respawn:/sbin/sulogin
```



## Ejemplo de inittab (2)



```
# what to do when CTRL-ALT-DEL is pressed
ca::ctrlaltdel:/sbin/shutdown -r -t 4 now
```

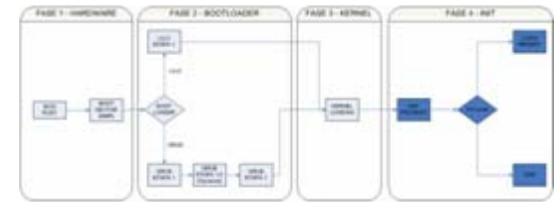
```
# special keyboard request (Alt-UpArrow)
# look into the kbd-0.90 docs for this
kb::kbrequest:/bin/echo "Keyboard Request -- edit /etc/inittab to
  let this work."
```

```
# what to do when power fails/returns
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
#pn::powerfail:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
```

```
# getty-programs for the normal runlevels
# <id>:<runlevels>:<action>:<process>
# The "id" field MUST be the same as the last
# characters of the device (after "tty").
1:2345:respawn:/sbin/mingetty --noclear tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```



# Runlevels (System V)

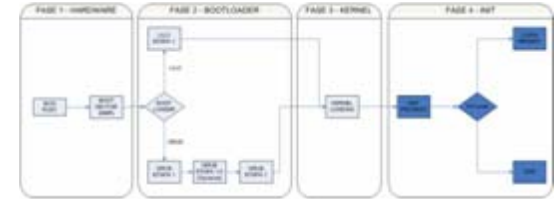


- 0 – Halt
- 1 – Single-user modo texto
- 2 – (Definible por el usuario)
- 3 – Multi-user completo modo texto
- 4 – (Definible por el usuario)
- 5 – Multi-user completo modo X-Windows
- 6 – Reboot (rearranque)





# rc.sysinit

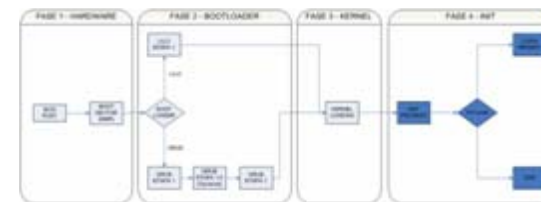


Este fichero varía según la distribución que utilicemos, parte puede estar incluida en los scripts rc (runlevel scripts) e incluso puede presentarse con otros nombres, pero en general realiza las tareas siguientes:

1. Configura el reloj del sistema en base al reloj hardware
2. Asigna el mapeo de teclado para la consola
3. Monta el sistema de ficheros /proc
4. Arranca el espacio de swap
5. Monta y chequea los sistemas de ficheros locales (no de red)
6. Inicializa el árbol de dependencias de módulos (depmod). (Esto permitirá la posterior utilización de modprobe para cargar los módulos)
7. Inicializa y configura las interfaces de red.
8. Carga los drivers para USB, PCMCIA, sonido.



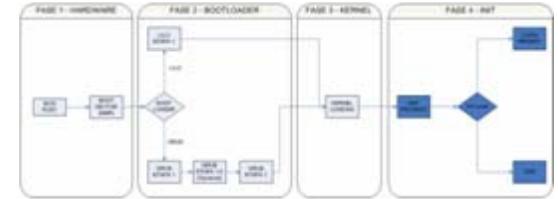
## Runlevel scripts (scripts rc)



- El script rc ejecuta los scripts de arranque que se encuentran en el directorio correspondiente al runlevel en el que se ha iniciado el sistema.
- Normalmente estos se encuentran en el directorio `/etc/rc.d/rcN.d/` (siendo N el runlevel)  
Es decir para el runlevel 3: `/etc/rc.d/rc3.d`
- En este directorio se encuentran una serie de scripts que empiezan por “S” o por “K” y dos dígitos
- Los dígitos denotan el orden de ejecución de los scripts:
  - Los que comienzan por S se ejecutan en orden ascendente (al arrancar o comenzar un nuevo runlevel)
  - Los que comienzan por K se ejecutan en orden descendente al salir (normalmente al hacer shutdown o al finalizar un runlevel)
- En realidad es una convención, porque rc llama a estos scripts con el parámetro start o stop en función de si es arranque o parada (de hecho los scripts K\* son enlaces simbólicos a los S\*)
- Hay un script particular que es el S99local donde se deben hacer configuraciones locales del equipo (si es necesario), se trata de evitar tocar los scripts previos por razones de facilidad de mantenimiento.



# Ejemplo de /etc/rcN.d



```
ninakula:/etc/rc.d/rc3.d # ls
.                K10running-kernel  S01hotplug      S121dap
..               K10sshd            S01lisdn        S12running-kernel
K02splash_late  K12nfs             S01random       S12sshd
K03cron         K12nfsboot         S02coldplug     S13kbd
K03xinetd       K14portmap         S05network      S13powersaved
K05postfix      K14resmgr          S06syslog       S13splash
K06nscd         K14smbfs           S08portmap      S14hwscan
K07cups         K14splash_early   S08resmgr       S14slpd
K08hwscan      K14vmware          S08smbfs        S15cups
K08slpd        K16syslog          S08splash_early S16nscd
K09powersaved  K17network         S08vmware       S17postfix
K09splash      K20coldplug        S10nfs          S19cron
K10alsasound   K21hotplug         S10nfsboot      S19xinetd
K10fbset       K21lisdn           S12alsasound    S20splash_late
K10ldap        K21random          S12fbset        S99webmin
ninakula:/etc/rc.d/rc3.d #
```



# Referencias

- La Wikipedia: <http://www.wikipedia.org>
- Códigos de error de la BIOS:  
<http://www.pchell.com/hardware/beepcodes.shtml>
- Información sobre BIOS: <http://www.bioscentral.com>
- Linux BIOS: [http://www.linuxbios.org/index.php/Main\\_Page](http://www.linuxbios.org/index.php/Main_Page)
- Especificación “El torito”:  
<http://www.phoenix.com/NR/rdonlyres/98D3219C-9CC9-4DF5-B496-A286D893E36A/0/specscdrom.pdf>
- SysLinux: <http://syslinux.zytor.com>
- LILO Howto: <http://www.tldp.org/HOWTO/LILO.html>
- GRUB: <http://www.gnu.org/software/grub/grub.html>
- Kernel Linux: <http://www.kernel.org>
- The Linux System Administrator Guide (sag):  
<http://mirrors.kernel.org/LDP/guides.html>



# ¿Preguntas?



# APENDICES

- Códigos de error de la BIOS
  - IBM BIOS
  - AMI BIOS
  - Phoenix BIOS

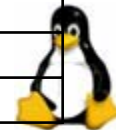


## Original IBM Error Codes

|                       |   |
|-----------------------|---|
| 1 short beep          | Normal POST - system is ok                      |
| 2 short beeps         | POST Error - error code shown on screen         |
| No beep               | Power supply or system board problem            |
| Continuou s beep      | Power supply, system board, or keyboard problem |
| Repeating short beeps | Power supply or system board problem            |
| 1 long, 1 short beep  | System board problem                            |
| 1 long, 2 short beeps | Display adapter problem (MDA, CGA)              |
| 1 long, 3 short beeps | Enhanced Graphics Adapter (EGA)                 |
| 3 long beeps          | 3270 keyboard card                              |

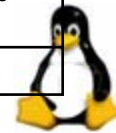
## IBM POST Diagnostic Code Descriptions

|               |   |
|---------------|---|
| 100 - 199     | System Board  |
| 200 - 299     | Memory  |
| 300 - 399     | Keyboard  |
| 400 - 499     | Monochrome Display                                  |
| 500 - 599     | Color/Graphics Display                              |
| 600 - 699     | Floppy-disk drive and/or Adapter                    |
| 700 - 799     | Math Coprocessor                                    |
| 900 - 999     | Parallel Printer Port                               |
| 1000 - 1099   | Alternate Printer Adapter                           |
| 1100 - 1299   | Asynchronous Communication Device, Adapter, or Port |
| 1300 - 1399   | Game Port   |
| 1400 - 1499   | Color/Graphics Printer                              |
| 1500 - 1599   | Synchronous Communication Device, Adapter, or Port  |
| 1700 - 1799   | Hard Drive and/or Adapter                           |
| 1800 - 1899   | Expansion Unit (XT)                                 |
| 2000 - 2199   | Bisynchronous Communication Adapter                 |
| 2400 - 2599   | EGA system-board Video (MCA)                        |
| 3000 - 3199   | LAN Adapter   |
| 4800 - 4999   | Internal Modem                                      |
| 7000 - 7099   | Phoenix BIOS Chips                                  |
| 7300 - 7399   | 3.5" Disk Drive                                     |
| 8900 - 8999   | MIDI Adapter  |
| 11200 - 11299 | SCSI Adapter  |
| 21000 - 21099 | SCSI Fixed Disk and Controller                      |
| 21500 - 21599 | SCSI CD-ROM System                                  |



# AMI BIOS Beep Codes

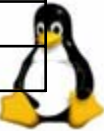
|                       |  |
|-----------------------|--|
| 1 Short Beep          | One beep is good! Everything is ok, that is if you see things on the screen. If you don't see anything, check your monitor and video card first. Is everything connected? If they seem fine, your motherboard has some bad chips on it. First reset the SIMM's and reboot. If it does the same thing, one of the memory chips on the motherboard are bad, and you most likely need to get another motherboard since these chips are soldered on.   |
| 2 Short Beeps         | Your computer has memory problems. First check video. If video is working, you'll see an error message. If not, you have a parity error in your first 64K of memory. First check your SIMM's. Reseat them and reboot. If this doesn't do it, the memory chips may be bad. You can try switching the first and second banks memory chips. First banks are the memory banks that your CPU finds its first 64K of base memory in. You'll need to consult your manual to see which bank is first. If all your memory tests good, you probably need to buy another motherboard. |
| 3 Short Beeps         | Basically the same thing as 2 beeps. Follow that diagnosis above.  |
| 4 Short Beeps         | Basically the same thing as 2 beeps. Follow that diagnosis above. It could also be a bad timer   |
| 5 Short Beeps         | Your motherboard is complaining. Try reseating the memory and rebooting. If that doesn't help, you should consider another motherboard. You could probably get away with just replacing the CPU, but that's not too cost-effective. Its just time to upgrade!  |
| 6 Short Beeps         | The chip on your motherboard that controls your keyboard (A20 gate) isn't working. First try another keyboard. If it doesn't help, reseate the chip that controls the keyboard, if it isn't soldered in. If it still beeps, replace the chip if possible. Replace the motherboard if it is soldered in.  |
| 7 Short Beeps         | Your CPU broke overnight. Its no good. Either replace the CPU, or buy another motherboard.   |
| 8 Short Beeps         | Your video card isn't working. Make sure it is seated well in the bus. If it still beeps, either the whole card is bad or the memory on it is. Best bet is to install another video card.  |
| 9 Short Beeps         | Your BIOS is bad. Reseat or Replace the BIOS.  |
| 10 Short Beeps        | Your problem lies deep inside the CMOS. All chips associated with the CMOS will likely have to be replaced. Your best bet is to get a new motherboard.   |
| 11 Short Beeps        | Your problem is in the Cache Memory chips on the motherboard. Reseat or Replace these chips.   |
| 1 Long, 3 Short Beeps | You've probably just added memory to the motherboard since this is a conventional or extended memory failure. Generally this is caused by a memory chip that is not seated properly. Reseat the memory chips.  |
| 1 Long, 8 Short Beeps | Display / retrace test failed. Reseat the video card.  |





# Phoenix BIOS Beep Codes (1)

|        |   |
|--------|---|
| 1-1-3  | Your computer can't read the configuration info stored in the CMOS. Replace the motherboard.  |
| 1-1-4  | Your BIOS needs to be replaced.   |
| 1-2-1  | You have a bad timer chip on the motherboard. You need a new motherboard.   |
| 1-2-2  | The motherboard is bad.   |
| 1-2-3  | The motherboard is bad.   |
| 1-3-1  | You'll need to replace the motherboard.   |
| 1-3-3  | You'll need to replace the motherboard.   |
| 1-3-4  | The motherboard is bad.   |
| 1-4-1  | The motherboard is bad.   |
| 1-4-2  | Some of your memory is bad.   |
| 2- _ _ | Any combo of beeps after two means that some of your memory is bad, and unless you want to get real technical, you should probably have the guys in the lab coats test the memory for you. Take it to the shop. |
| 3-1- _ | One of the chips on your motherboard is broken. You'll likely need to get another board.  |
| 3-2-4  | One of the chips on your motherboard that checks the keyboard is broken. You'll likely need to get another board.   |
| 3-3-4  | Your computer can't find the video card. Is it there? If so, try swapping it with another one and see if it works.  |
| 3-4- _ | Your video card isn't working. You'll need to replace it.   |
| 4-2-1  | There's a bad chip on the motherboard. You need to buy another board.   |
| 4-2-2  | First check the keyboard for problems. If nothing, you have a bad motherboard.  |
| 4-2-3  | Same as 4-2-2.  |
| 4-2-4  | One of the cards is bad. Try yanking out the cards one by one to isolate the culprit. Replace the bad one. The last possibility is to buy another motherboard.  |
| 4-3-1  | Replace the motherboard.  |
| 4-3-2  | See 4-3-1   |
| 4-3-3  | See 4-3-1   |



## Phoenix BIOS Beep Codes (y 2)

|           |   |
|-----------|---|
| 4-3-4     | Time of day clock failure. Try running the setup program that comes with the computer. Check the date and time. If that doesn't work, replace the battery. If that doesn't work, replace the power supply. You may have to replace the motherboard, but that is rare. |
| 4-4-1     | Your serial ports are acting up. Reseat, or replace, the I/O card. If the I/O is on the motherboard itself, disable them with a jumper (consult your manual to know which one) and then add an I/O card.  |
| 4-4-2     | See 4-4-1, but this time is your Parallel port that's acting up.  |
| 4-4-3     | You math coprocessor is having problems. Run a test program to double-check it. If it is indeed bad, disable it, or replace it.   |
| Low 1-1-2 | Your motherboard is having problems   |
| Low 1-1-3 | This is an Extended CMOS RAM problem, check your motherboard battery, and motherboard.  |

- **These audio codes are a little more detailed than the AMI codes.**
- **This BIOS emits three sets of beeps. For example, 1 -pause- 3 -pause 3 -pause. This is a 1-3-3 combo and each set of beeps is separated by a brief pause.**
- **Listen to this sequence of sounds, count them, and reboot and count again if you have to.**



# Guide of recovering GRUB after reinstalling Windows

- Make sure to note down your GRUB configuration.

```
# Note that you do not have to rerun grub after making changes to this file
# NOTICE: You do not have a /boot partition. This means that
#     all kernel and initrd paths are relative to /, eg.
#     root (hd0,2)
#     kernel /boot/vmlinuz-version ro root=/dev/hda3
#     initrd /boot/initrd-version.img
#boot=/dev/hda
default=0
timeout=5
#hiddenmenu
splashimage=(hd0,2)/boot/grub/splash.xpm.gz
title Windows XP
    rootnoverify (hd0,0)
    chainloader +1
title Fedora Core (2.6.9-1.667)
    root (hd0,2)
    kernel /boot/vmlinuz-2.6.9-1.667 ro root=LABEL=/ rhgb quiet vga=792
    initrd /boot/initrd-2.6.9-1.667.img
```

- After installing Windows os, booting your computer with the first disk of your Fedora Core installation disks.
- When see the installation UI of FC3, choose F4 or type "linux rescue". After the linux root prompt is display in the screen:

```
sh#grub
grub>root(hd0,2)
grub>setup (hd0)
```

- root (hd0,2) means to set your linux root to (hd0,2) "Second partition of your first disk of your computer ". it should comply with the setting in grub.conf . setup (hd0) tells Grub to rewrite MBR of your computer. after grub had done successfully, the screen will prompt "Successful....". Then it's safe to reboot and your GRUB will reappear.

