

# Inicio del sistema

Gunnar Wolf



# Índice

- 1 Introducción
- 2 La carga inicial
- 3 Reconociendo el entorno
- 4 Inicio del espacio de usuario



# Enfocada a lo específico

- Voy a enfocar esta presentación al inicio de un sistema Linux
  - Nos da el mayor detalle acerca del proceso
  - Guarda bitácora de toda la información de inicio
  - Permite *implementaciones competitivas* de subsistemas (y permite un debate amplio, rico al respecto)



# ¿Es tan difícil?

- La arquitectura esquemática básica de una computadora es bastante trivial
- Cargar un sistema operativo... Tal vez no tanto
- Nos desviaremos brevemente del programa formal de la materia para comprender qué es lo que ocurre *en los primeros instantes*
  - Buscando entender hasta el punto en que tenemos un sistema *usable*



# Índice

- 1 Introducción
- 2 La carga inicial**
- 3 Reconociendo el entorno
- 4 Inicio del espacio de usuario



# Encendiendo la computadora: Alcance del BIOS

- Verificación de *sanidad* del sistema (POST)
  - North Bridge CPU+RAM (arquitectura von Neumann)
  - South Bridge Dispositivos, componentes adicionales (USB, PCI, red, SATA, etc.)
- Enumeración básica de dispositivos
- Selección del dispositivo de arranque
- Carga del *primer sector* del dispositivo seleccionado
  - Verificación de *integridad* del código a ejecutar (lo veremos a detalle la siguiente sesión)
  - Transferencia de control a los elementos *provistos por el usuario* (sistema operativo)



## ¿Por qué tenemos programas *cargadores*?

- En arquitecturas derivadas de Intel, el BIOS opera en *modo real*
  - Esto es, con un límite de 640Kb RAM, según la especificación de la PC de 1981
  - En años recientes, se va migrando de BIOS a EFI → Veremos la próxima clase lo que eso conlleva
- El conocimiento del BIOS del sistema es muy limitado
  - Su especificación no sabe más que pedir *el primer sector* (512 bytes) del disco duro
  - Cualquier programa que cargue, debe caber en 512 bytes
  - De ahí que tengamos *cargadores de inicio* (*boot loaders*)







# Los cargadores de inicio

- Prácticamente cualquier sistema operativo moderno depende de un cargador de inicio
- Estos han ido creciendo para convertirse en verdaderos mini-sistemas operativos
  - Particularmente en el área de sistemas de archivo y de E/S
  - Enumeración de dispositivos (no siempre *heredado* del BIOS)
  - Comprensión de distintos sistemas de archivos (incluso abstracciones como RAID/LVM)
  - Análisis del estado de la última carga (sugiriendo *modo a prueba de fallos*)
  - Edición de los parámetros de invocación del kernel



## El trabajo del cargador

Por fin, el trabajo del cargador típicamente se reduce a:

- Informar al usuario que *todo va bien*
- Reconocer el entorno
- Ubicar y cargar la imagen del sistema operativo en el medio de arranque
  - Posiblemente también de un *disco de inicio mínimo*
- Especificar parámetros de inicio
- Transferir la ejecución (y *suicidarse*)

... Y entramos en el terreno del sistema operativo



# Índice

- 1 Introducción
- 2 La carga inicial
- 3 Reconociendo el entorno**
- 4 Inicio del espacio de usuario



# La fuente de nuestros datos

- Bitácora en `/var/log/dmesg`
- La bitácora tiene un *timestamp* en cada línea de su bitácora, con resolución de microsegundos
  - Eliminada de lo que aquí muestro para que quepa mejor en pantalla



## Primeros pasos: ¿Dónde estoy?

¿La arquitectura es capaz de correr el núcleo en cuestión?

```
Linux version 3.2.0-4-amd64 (debian-kernel@lists.debian.org)
(gcc version 4.6.3 (Debian 4.6.3-14) ) #1 SMP Debian
3.2.35-2
Command line: BOOT_IMAGE=/vmlinuz-3.2.0-4-amd64
root=/dev/mapper/mosca-root ro vga=791 quiet splash
```

¿Cuál es el *mapa de memoria*?

BIOS-provided physical RAM map:

```
BIOS-e820: 0000000000000000 - 000000000009ec00 (usable)
BIOS-e820: 00000000000f0000 - 0000000000100000 (reserved)
BIOS-e820: 0000000000100000 - 00000000cd9ffc00 (usable)
BIOS-e820: 00000000cd9ffc00 - 00000000cda53c00 (ACPI NVS)
BIOS-e820: 00000000cda53c00 - 00000000cda55c00 (ACPI data)
BIOS-e820: 00000000cda55c00 - 00000000d0000000 (reserved)
(...)
BIOS-e820: 00000000ffb00000 - 0000000100000000 (reserved)
BIOS-e820: 0000000100000000 - 0000000128000000 (usable)
```



## Características base del equipo

Sigue descubriendo información sobre el CPU y la memoria

```
NX (Execute Disable) protection: active
(...)
found SMP MP-table at [ffff880000fe710] fe710
ACPI: XSDT 0000000000fc7f0 0008C (v01 DELL B10K 00000015
ASL 00000061)
(...)
No NUMA configuration found
Faking a node at 0000000000000000-0000000128000000
Initmem setup node 0 0000000000000000-0000000128000000
(...)
ACPI: IRQ0 used by override.
ACPI: IRQ2 used by override.
ACPI: IRQ9 used by override.
Using ACPI (MADT) for SMP configuration information
ACPI: HPET id: 0x8086a701 base: 0xfed00000
SMP: Allowing 8 CPUs, 6 hotplug CPUs
(...)
Booting paravirtualized kernel on bare hardware
```



## Inicia la ejecución y recorrida de los *buses*

```
Memory: 3880132k/4849664k available (3418k kernel code, 825804k
absent, 143728k reserved, 3319k data, 576k init)
Hierarchical RCU implementation.
RCU dyntick-idle grace-period acceleration is enabled.
NR_IRQS:33024 nr_irqs:744 16
Console: colour dummy device 80x25
console [tty0] enabled
hpet clockevent registered
Fast TSC calibration using PIT
Detected 2992.557 MHz processor.
```

- Hasta este punto, *todo ocurre con  $t=0.000000$*
- No es que sea un proceso *tan* instantáneo, sino que el kernel no ha comenzado a registrar el paso del tiempo

```
Calibrating delay loop (skipped), value calculated using timer
frequency.. 5985.11 BogoMIPS (lpj=11970228)
```

- Y acá el tiempo inicia.



# Características de ejecución de procesos en sistema

## Límites y subsistemas de control

```
pid_max: default: 32768 minimum: 301
Security Framework initialized
AppArmor: AppArmor disabled by boot time parameter
Dentry cache hash table entries: 524288 (order: 10, 4194304
bytes)
Inode-cache hash table entries: 262144 (order: 9, 2097152 bytes)
Mount-cache hash table entries: 256
Initializing cgroup subsys cpuacct
Initializing cgroup subsys memory
Initializing cgroup subsys devices
Initializing cgroup subsys freezer
Initializing cgroup subsys net_cls
Initializing cgroup subsys blkio
Initializing cgroup subsys perf_event
```





## Vamos subiendo de nivel: Dispositivos base

Fundamental para multitarea: Cómo hablar con el temporizador,  
manejo de interrupciones

```
..TIMER: vector=0x30 apic1=0 pin1=2 apic2=-1 pin2=-1  
(...)  
NMI watchdog enabled, takes one hw-pmu counter.  
Brought up 2 CPUs  
Total of 2 processors activated (12011.18 BogoMIPS).
```

Y aquí (0.29s) comienzan a activarse los subsistemas que empleará  
el usuario

```
devtmpfs: initialized  
(...)  
NET: Registered protocol family 16  
(...)  
[Firmware Bug]: ACPI: BIOS _OSI(Linux) query ignored  
ACPI: Interpreter enabled  
ACPI: (supports S0 S1 S3 S4 S5)
```



## Comienza enumeración de dispositivos

Primer bus en ser *barrido*: PCI. Proceso largo (0.46s–1.9s)

```
PCI: Using host bridge windows from ACPI; if necessary, use
"pci=nocrs" and report a bug
ACPI: PCI Root Bridge [PCI0] (domain 0000 [bus 00-ff])
pci_root PNP0A03:00: host bridge window [io 0x0000-0x0cf7]
pci_root PNP0A03:00: host bridge window [io 0x0d00-0xffff]
pci_root PNP0A03:00: host bridge window [mem
0x000a0000-0x000bffff]
pci_root PNP0A03:00: host bridge window [mem
0x000c0000-0x000effff]
(...)
pci 0000:00:00.0: [8086:2e10] type 0 class 0x000600
pci 0000:00:01.0: [8086:2e11] type 1 class 0x000604
pci 0000:00:01.0: PME# supported from D0 D3hot D3cold
(...)
ACPI: PCI Interrupt Link [LNKA] (IRQs 3 4 5 6 7 9 10 *11 12 15)
ACPI: PCI Interrupt Link [LNKB] (IRQs 3 4 *5 6 7 9 10 11 12 15)
ACPI: PCI Interrupt Link [LNKC] (IRQs 3 4 5 6 7 *9 10 11 12 15)
```



## Se reservan *puertos* de memoria → dispositivos

Como veremos posteriormente, para *hablar* con ciertos dispositivos lo haremos a través de *regiones de memoria* dedicadas a entrada/salida

```
ACPI: bus type pnp registered
pnp 00:00: [bus 00-ff]
pnp 00:00: [io 0x0cf8-0x0cff]
pnp 00:00: [io 0x0000-0x0cf7 window]
pnp 00:00: [io 0x0d00-0xffff window]
pnp 00:00: [mem 0x000a0000-0x000bffff window]
pnp 00:00: [mem 0x000c0000-0x000effff window]
(...)
pci 0000:00:1f.2: BAR 5: assigned [mem 0xf0000000-0xf00007ff]
pci 0000:00:03.0: BAR 0: assigned [mem 0xf0000800-0xf000080f
64bit]
pci 0000:00:1c.1: BAR 15: assigned [mem 0xf0100000-0xf02fffff
64bit pref]
```



## Encontramos conexiones con otros buses

Vemos que el bus PCI es el *maestro*, y de él descienden otros varios buses

```
pci 0000:00:01.0: PCI bridge to [bus 01-01]
pci 0000:00:01.0:   bridge window [mem 0xfe500000-0xfe5ffffff]
pci 0000:00:1c.0: PCI bridge to [bus 02-02]
pci 0000:00:1c.0:   bridge window [io 0x2000-0x2fff]
pci 0000:00:1c.0:   bridge window [mem 0xfe400000-0xfe4ffffff]
pci 0000:00:1c.0:   bridge window [mem 0xf0300000-0xf04ffffff
64bit pref]
pci 0000:00:1c.1: PCI bridge to [bus 03-03]
pci 0000:00:1c.1:   bridge window [io 0x1000-0x1fff]
pci 0000:00:1c.1:   bridge window [mem 0xfe300000-0xfe3ffffff]
pci 0000:00:1c.1:   bridge window [mem 0xf0100000-0xf02ffffff
64bit pref]
pci 0000:00:1e.0: PCI bridge to [bus 04-04]
pci 0000:00:01.0: setting latency timer to 64
pci 0000:00:1c.0: setting latency timer to 64
pci 0000:00:1c.1: setting latency timer to 64
pci 0000:00:1e.0: setting latency timer to 64
```



## Últimos toques antes de montar un micro-sistema

```
NET: Registered protocol family 2
IP route cache hash table entries: 131072 (order: 8, 1048576
  bytes)
TCP established hash table entries: 524288 (order: 11, 8388608
  bytes)
TCP bind hash table entries: 65536 (order: 8, 1048576 bytes)
TCP: Hash tables configured (established 524288 bind 65536)
TCP reno registered
UDP hash table entries: 2048 (order: 4, 65536 bytes)
UDP-Lite hash table entries: 2048 (order: 4, 65536 bytes)
NET: Registered protocol family 1
pci 0000:00:02.0: Boot video device
PCI: CLS 64 bytes, default 64
Unpacking initramfs...
Freeing initrd memory: 10904k freed
```

- Con un *ramdisk de inicio* montado termina la *fase inicial* de la carga.
- 1.98s → 2.17s
- El núcleo sigue reconociendo su entorno...



## Estructuras de gestión de memoria

- Veremos el rol de las *tablas de traducción de direcciones* (Translation Lookaside Buffer, TLB) en la unidad *Administración de memoria*

```
Placing 64MB software IO TLB between ffff8800c99fd000 -  
ffff8800cd9fd000  
software IO TLB at phys 0xc99fd000 - 0xcd9fd000  
Simple Boot Flag at 0x7a set to 0x1  
audit: initializing netlink socket (disabled)  
type=2000 audit(1362187934.168:1): initialized  
HugeTLB registered 2 MB page size, pre-allocated 0 pages
```



## Comienzan las cargas de módulos controladores

- Llegamos aquí a los 2.18s
- Comienza la parte más lenta del inicio
  - La inicialización de cada dispositivo tarda en promedio entre  $\frac{1}{100}$ s y  $\frac{1}{10}$ s

```
VFS: Disk quotas dquot_6.5.2
Block layer SCSI generic (bsg) driver version 0.4 loaded (major
253)
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
(...)
vesafb: mode is 1024x768x16, linelength=2048, pages=0
vesafb: framebuffer at 0xd0000000, mapped to
0xffffc90011100000, using 1536k, total 1536k
Console: switching to colour frame buffer device 128x48
(...)
Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
serial8250: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
00:07: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
```



## Me parece que ya hay hilos dentro del núcleo

- Vemos que se van inicializando elementos con poca relación entre sí
- Indicio de hilos que van avanzando en paralelo

```
mousedev: PS/2 mouse device common for all mice
rtc_cmos 00:05: RTC can wake from S4
rtc_cmos 00:05: rtc core: registered rtc_cmos as rtc0
rtc0: alarms up to one day, 242 bytes nvram, hpet irqs
cpuidle: using governor ladder
cpuidle: using governor menu
TCP cubic registered
NET: Registered protocol family 10
Mobile IPv6
NET: Registered protocol family 17
Registering the dns_resolver key type
(...)
rtc_cmos 00:05: setting system clock to 2013-03-02 01:32:15 UTC
(1362187935)
Freeing unused kernel memory: 576k freed
Write protecting the kernel read-only data: 6144k
```





## Más subsistemas: UDev, USBcore, EHCI, ATA...

```
udev[52]: starting version 175
(...)
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
libata version 3.00 loaded.
uhci_hcd: USB Universal Host Controller Interface driver
(...)
usb usb1: New USB device found, idVendor=1d6b, idProduct=0002
usb usb1: New USB device strings: Mfr=3, Product=2,
  SerialNumber=1
usb usb1: Product: EHCI Host Controller
usb usb1: Manufacturer: Linux 3.2.0-4-amd64 ehci_hcd
usb usb1: SerialNumber: 0000:00:1a.7
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 6 ports detected
ata_generic 0000:00:03.2: setting latency timer to 64
scsi0 : ata_generic
scsi1 : ata_generic
atal: PATA max UDMA/100 cmd 0xfe80 ctl 0xfe90 bmdma 0xfef0 irq
```



## Caminando los buses

- A partir de este punto, el núcleo averigua lo que tiene conectado por el bus USB
- Líneas bastante repetitivas
- Desde los 2.64 hasta los 3.0 segundos
- Comienzan a aparecer los discos SATA/SCSI (por libata)



## Caminando los buses: SATA

```
ata8: SATA link down (SStatus 0 SControl 300)
ata6: SATA link up 1.5 Gbps (SStatus 113 SControl 300)
ata4: SATA link down (SStatus 0 SControl 300)
ata5: SATA link up 1.5 Gbps (SStatus 113 SControl 300)
ata3: SATA link up 3.0 Gbps (SStatus 123 SControl 300)
(...)
scsi 2:0:0:0: Direct-Access      ATA          WDC WD5000AAKS-2 12.0
                PQ: 0 ANSI: 5
scsi 4:0:0:0: CD-ROM             TSSTcorp    DVD+-RW TS-H653G DW10
                PQ: 0 ANSI: 5
sd 2:0:0:0: [sda] 976773168 512-byte logical blocks: (500
                GB/465 GiB)
sd 2:0:0:0: [sda] Write Protect is off
sd 2:0:0:0: [sda] Mode Sense: 00 3a 00 00
sd 2:0:0:0: [sda] Write cache: enabled, read cache: enabled,
                doesn't support DPO or FUA
sr0: scsi3-mmc drive: 48x/48x writer dvd-ram cd/rw xa/form2
                cdda tray
```



## Por fin: Sistemas de archivos

- Al cargar los controladores de sistemas de archivos (3.55s), ya podemos montar y comenzar a iniciar el *entorno operativo*
- Esto sigue *sumergido* entre mensajes de descubrimiento de dispositivos (principalmente unidades de disco y dispositivos USB)
- Veremos posteriormente características de ReiserFS y de otros sistemas de archivos

```
REISERFS (device dm-0): found reiserfs format "3.6" with
    standard journal
REISERFS (device dm-0): using ordered data mode
reiserfs: using flush barriers
REISERFS (device dm-0): journal params: device dm-0, size 8192,
    journal first block 18, max trans len 1024, max batch 900,
    max commit age 30, max trans age 30
REISERFS (device dm-0): checking transaction log (dm-0)
REISERFS (device dm-0): Using r5 hash to sort names
```



## Algunos puntos sueltos. . .

- La inicialización continúa
- En este caso, *caminar* los buses USB llevó hasta los 5.06s
- *Silencio* de 11.4s después de que inicia `udev` [347]
  - Puede estar estructurando lo encontrado en el sistema `/dev` dinámico (?)
- Otras líneas interesantes (sueltas):

```
input: PC Speaker as /devices/platform/pcspkr/input/input0
input: Power Button as
      /devices/LNXSYSTM:00/device:00/PNP0C0C:00/input/input1
Adding 2097148k swap on /dev/mapper/mosca-swap. Priority:-1
      extents:1 across:2097148k
EXT4-fs (sda2): mounted filesystem with ordered data mode.
      Opts: (null)
kjournald starting. Commit interval 5 seconds
```



# Índice

- 1 Introducción
- 2 La carga inicial
- 3 Reconociendo el entorno
- 4 Inicio del espacio de usuario**



# ¡Bienvenido a tu sistema!

- Tras este punto, estamos inequívocamente ya corriendo en un sistema completo
- La primer evidencia de ello: La carga de `udev` como proceso  
`> pid_min (301)` a los 5.06s
- Ahora, ¿qué sigue?



## El primer proceso: `init`

- En un sistema Unix, tan pronto el núcleo está listo para ejecutar *algo* ejecuta al proceso `/sbin/init`
- `init` se encarga de inicializar al resto del sistema
  - ¿Cómo?
- Estrategias *clásicas*: Sistemas BSD, SysV
  - Todos los sistemas tipo Unix, hasta hace unos cinco años siguen alguna de estas dos vías
- Estrategias *modernas*:
  - Orientado a *eventos*: `upstart`
  - Orientado a *sockets*: `systemd`





## Esquema BSD (1)

`init` sigue una serie de simples *scripts* para sus principales operaciones (de: Gestión del sistema OpenBSD):

- `/etc/rc` Fichero de configuración principal. No se debe editar.
- `/etc/rc.conf` Fichero de configuración usado por `/etc/rc` para saber qué `dæmon` deben iniciarse con el sistema.
- `/etc/rc.conf.local` Fichero de configuración que se puede usar para anular configuraciones de `/etc/rc.conf`, sin que sea necesario tocar el fichero `/etc/rc.conf`; muy conveniente para los usuarios que actualizan el sistema con frecuencia.

(continúa)



## Esquema BSD (2)

- `/etc/netstart` Fichero de configuración usado para iniciar la red. No se debe editar.
- `/etc/rc.local` Fichero de configuración usado para tareas de administración local. Aquí se debe almacenar información específica de la máquina anfitriona o de los `dæmon`.
- `/etc/rc.securelevel` Fichero de configuración para ejecutar las órdenes que deben ser invocadas antes de los cambios en el nivel de seguridad. Véase `init(8)`.
- `/etc/rc.shutdown` Fichero de configuración invocado por `shutdown(8)`. En este fichero se debe añadir cualquier cosa que se quiera hacer antes de cerrar el sistema. Véase `rc.shutdown(8)`.



# Esquema SysV (1)

- Parte de la idea de diferentes *conjuntos de programas* que pueden ser requeridos por distintos *usos del sistema (runlevels)*
- Busca menor dependencia en el administrador para indicar qué se va a iniciar
  - Permitiendo que los distintos *paquetes* se ubiquen en el punto correcto
- Configuración basada en un archivo base, `/etc/inittab`, y un directorio por runlevel (`/etc/rc0.d/` a `/etc/rc6.d/`)



## Esquema SysV (2)

- En cada directorio se colocan archivos (ejecutados en orden *alfabético*) indicando si *inician* (S) o *detienen* (K) al servicio
  - Esta convención apunta a que p.ej. `/etc/rc3.d/S20gdm3` inicia `gdm3` al estar en runlevel 3, y con un *ordenamiento* de 20
- Los runlevels *no son acumulativos* (no hay relación que un nivel *contenga* al anterior)
- Algunos *runlevels* tienen significado especial
  - 0 Shutdown (apagado)
  - 1 Mantenimiento / monousuario
  - 6 Restart (reinicio)

