

# Programación y Administración de Sistemas

## 3. Arranque y parada del sistema

Pedro Antonio Gutiérrez

Asignatura "Programación y Administración de Sistemas"  
2º Curso Grado en Ingeniería Informática  
Escuela Politécnica Superior  
(Universidad de Córdoba)  
[pagutierrez@uco.es](mailto:pagutierrez@uco.es)

22 de febrero de 2017



# Objetivos del aprendizaje I

- Describir todas las fases del proceso de arranque del sistema GNU/Linux, estableciendo qué elementos *hardware* y, sobre todo, *software* intervienen.
- Explicar qué misión tienen en el arranque los siguientes componentes: iniciador ROM, programa cargador, núcleo del sistema operativo, `initrd` y proceso `Init`.
- Configurar los distintos parámetros del programa cargador GRUB y utilizar el modo interactivo.
- Distinguir la diferencia entre modo de ejecución monousuario y multiusuario.
- Explicar los problemas de seguridad asociados al modo monousuario.
- Enumerar los pasos necesario para la configuración del sistema en modo multiusuario.

# Objetivos del aprendizaje II

- Explicar el concepto de niveles de ejecución y las funciones típicas de cada nivel.
- Configurar los servicios de cada nivel de ejecución utilizando las carpetas `/etc/rc?.d/`.
- Arrancar, parar, listar, añadir y eliminar servicios en cada nivel.
- Configurar los servicios al inicio utilizando el sistema *Upstart*.
- Enumerar las acciones que se llevan a cabo durante la parada del sistema.
- Utilizar la herramienta `shutdown`.
- Enumerar posibles causas de caídas del sistema operativo.
- Enumerar posibles problemas durante el arranque del sistema operativo.
- Conocer los principales mecanismos para consultar los *logs* del sistema operativo.

## 3.1. Introducción.

## 3.2. Proceso de arranque del sistema.

### 3.2.1. Proceso de arranque.

3.2.1.1. Iniciador ROM.

3.2.1.2. Programa cargador.

3.2.1.3. Núcleo del sistema operativo.

3.2.1.4. initrd.

3.2.1.5. Proceso Init.

### 3.2.2. Programa cargador GRUB (GRand Unified Bootloader).

3.2.2.1. Ficheros de configuración.

3.2.2.2. Consola interactiva GRUB.

### 3.2.3. Modos monousuario/multiusuario.

3.2.3.1. Modo monousuario: concepto y problemas de seguridad.

3.2.3.2. Modo multiusuario: pasos del proceso de arranque.

### 3.2.4. Niveles de ejecución.

### 3.2.5. Ficheros de inicialización.

3.2.5.1. Carpetas /etc/rc?.d/.

- 3.2.5.2. Herramientas de arranque, parada, listado, adición y eliminación de servicios.

- 3.2.6. Upstart.

- 3.2.6.1. Objetivo y organización.

- 3.2.6.2. Ficheros de configuración de eventos.

- 3.3. Parada del sistema.

- 3.3.1. Acciones durante el proceso de parada.

- 3.3.2. Herramienta `shutdown`.

- 3.4. Caídas del sistema y problemas de arranque.

- 3.4.1. Posibles causas de caídas del sistema.

- 3.4.2. Problemas de arranque.

- 3.4.3. Ficheros de *log*.

- Cuestionarios objetivos.
- Pruebas de respuesta libre.
- Tareas de administración.

# Arranque y parada del sistema

## Procesos de **arranque** y de **parada**

- **Arranque**: el sistema se prepara para ser usado por los usuarios.
- **Parada**: el sistema se deja consistente (p.ej. vaciar la caché).
- El **administrador** deberá saber qué ficheros controlan estos procesos y cómo lo hacen, para reconocer situaciones de error y solucionarlas.
- Procesos **sencillos**: se basan en un conjunto de ficheros de configuración y de guiones *shell* que determinan y controlan los procesos.



# Proceso de arranque

- Dos fases:
  - Arranque del *hardware*.
  - Arranque del Sistema Operativo (SO).

Bajo el control del  
iniciador ROM

- Test del hardware
- Carga en memoria del cargador del SO

Bajo el control del  
cargador (*boof*) del SO

- Carga en memoria componentes del SO

Inicialización bajo el  
control de la parte  
residente del SO

- Test del sistema de ficheros
- Creación de estructuras de datos internas
- Completa la carga del SO residente
- Creación de procesos *login*

Se entra en la fase normal de funcionamiento del SO





## Proceso de arranque: iniciador ROM

- Al arrancar el ordenador  $\Rightarrow$  señal eléctrica (RESET) que inicializa todos los registros a valores por defecto.
- Se carga la dirección de inicio del **iniciador ROM**.
- La memoria ROM contiene, además, el *software* de configuración del *hardware* del sistema (BIOS).

### Iniciador ROM

Programa de arranque independiente del SO (ROM). 3 funciones:

- Comprueba el sistema, detectando sus características y comprobando su funcionamiento.
- Lee y almacena en memoria el **programa cargador** del SO.
- Pasa el control al cargador del SO, saltando a la dirección de memoria donde lo ha almacenado.





## Proceso de arranque: núcleo del S.O.

- El núcleo del S.O. continúa el proceso de arranque:
  - Realiza una comprobación del *hardware* del sistema.
  - Se prepara a sí mismo para ejecutar el sistema inicializando sus tablas internas, creando estructuras de datos necesarias, etc.
  - A continuación crea el proceso `Init` y le pasa el control.
- El núcleo ([Linux](#)) es cargado inicialmente en memoria, y permanece de manera residente durante el funcionamiento del sistema, controlando la ejecución del resto de *software* ([GNU](#)).
- Parte de este código se encuentra en módulos del núcleo:
  - Minimizar la cantidad de código que se carga en memoria.
  - Maximizar la modularidad.



# Proceso de arranque: `initrd`

- **`initrd`** (*Init RAM Disk*):

- Las características del arranque pueden implicar que el medio desde el que se carga el núcleo provenga de un sistema de ficheros concreto (p.ej. `ext3`) o incluso desde la red.
- Para ello, se necesitarán módulos específicos, alojados en el `initrd`.
- El programa cargador le dice al núcleo la posición del `initrd`.
- `initrd` evolucionó a `initramfs`, permitiendo un tamaño de disco variable y no ser necesario hacer un disco virtual para el núcleo.
- Funcionamiento:
  - El núcleo carga primero el `initrd`.
  - Utilizando el `initrd`, se cargan los módulos necesarios.
  - Entonces el núcleo continuará el proceso de arranque.

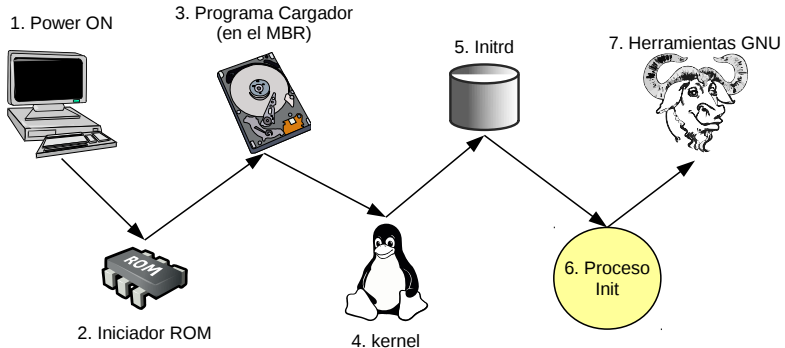


## Proceso de arranque: proceso `Init`

- El proceso `Init` termina el proceso de arranque, dejando el sistema en modo multiusuario, preparado para que los usuarios trabajen en él.
- Usa una serie de *scripts* que le indican las acciones a realizar.
- Tareas que realiza el proceso `Init`:
  - Chequea los sistemas de ficheros.
  - Monta los sistemas de ficheros permanentes.
  - Activa las áreas de *swapping* o intercambio.
  - Activa los demonios y la red (NFS, NIS, etc.).
  - Limpia los sistemas de ficheros (borra los directorios temporales).
  - Habilita el login a los usuarios del sistema.



# Proceso de arranque



# Gestor de arranque GRUB

- GRUB: GRand Unified Bootloader:

- GRUB se instala en el *master boot record* (MBR) y hace de las funciones de *master boot program* (MBP, programa cargador).
- Pregunta qué SO arrancar: p.ej. Linux o Windows.
  - Si la respuesta es Linux  $\Rightarrow$  carga el núcleo solicitado y le pasa el control para que el arranque continúe.
  - Si la respuesta es Windows  $\Rightarrow$  pasa el control a Windows que realiza su arranque.
- GRUB 2.0: desde el año 2009.
  - Archivo fundamental de configuración: `/boot/grub/grub.cfg`
  - ¡No editar a mano!.
  - Este archivo se genera a partir del comando `sudo update-grub2`, utilizando todos los *scripts* incluidos en la carpeta `/etc/grub.d/`.



# Gestor de arranque GRUB

- Contenidos de la carpeta `/etc/grub.d/`:
  - `/etc/grub.d/00_header`: Cabeceras, no se suele modificar.
  - `/etc/grub.d/05_debian_theme`: Aspecto visual del menú: colores, temas, imagen de fondo...
  - `/etc/grub.d/10_linux`: Este archivo contiene comandos y *scripts* que se encargan del *kernel* Linux de la partición principal (se incluyen todos los núcleos presentes en `/boot`).
  - `/etc/grub.d/20_*`: Aplicaciones *third party* (`20_memtest86+`, `20_linux_xen...`)
  - `/etc/grub.d/30_os-prober`: Este archivo contiene comandos y *scripts* que se encargan de otros sistemas operativos.
    - 4 secciones: Windows, otras particiones Linux, OSX y Hurd.
    - Los cambios que realicemos en una sección no afectarán al resto de las secciones.





# Gestor de arranque GRUB

- Fichero `/etc/default/grub`:
  - Este fichero si es editable (`00_header` lee su contenido).
  - `GRUB_DEFAULT=0`: entrada por defecto para el arranque. Si ponemos `saved`, será seleccionada por el administrador (comandos `grub-set-default`, `permanente`, y `grub-reboot`, un solo arranque).
  - `GRUB_SAVEDefault=true`: la entrada por defecto es siempre la última seleccionada.
  - `GRUB_HIDDEN_TIMEOUT=0`:
    - Muestra una pantalla en negro o con una imagen, durante el número de segundos indicado, antes del menú de arranque (pulsar una tecla para saltarla).
    - Suele no usarse cuando hay múltiples sistemas (comentado).
    - Es 0 cuando solo hay linux (el menú puede aparecer con `Shift`).
  - `GRUB_HIDDEN_TIMEOUT_QUIET=true`: sin cuenta atrás.



# Gestor de arranque GRUB

- Fichero `/etc/default/grub`:
  - `GRUB_TIMEOUT=10`: número de segundos hasta seleccionar entrada por defecto.
  - `GRUB_DISTRIBUTOR=\lsb_release -i -s 2> /dev/null || echo Debian`: obtener el nombre de la distribución.
  - `GRUB_CMDLINE_LINUX="opciones"`: pasar opciones de arranque al kernel linux (modo normal o recuperación).
  - `GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"`: pasar opciones de arranque al kernel linux (modo normal).
  - `GRUB_TERMINAL=console`: desactivar modo gráfico.



# Gestor de arranque GRUB

- Fichero `/etc/default/grub`:
  - `GRUB_DISABLE_LINUX_UUID="true"`: no utilizar el UUID del dispositivo raíz (utilizar nomenclatura tradicional `/dev/sda`).
  - `GRUB_GFXMODE=640x480`: seleccionar manualmente la resolución para el menú.
  - `GRUB_INIT_TUNE="480 440 1"`: hacer beep antes del menú de inicio (`tempo [pitch1 duration1] [pitch2 duration2]...`).
  - `GRUB_BACKGROUND`: imagen de fondo.
- Reinstalar GRUB (por ejemplo, después de que Windows borre el MBR): `sudo grub-install /dev/sda`.



# Gestor de arranque GRUB

- GRUB permite (durante la selección del SO):
  - Editar las entradas:
    - Pulsar tecla **e**, permite modificar las entradas de arranque para solucionar errores.
    - Los cambios no son permanentes, solo sirve para probar.
  - Consola interactiva GRUB: pulsar la tecla **c**. Permite ejecutar comandos para arreglar el arranque (seleccionar otro `initrd`, cargar módulos...).
  - Terminología de GRUB, numerando los dispositivos según los reconozca la BIOS empezando en cero:
    - Nombres de dispositivos: (`<t><n>`, `<np>`)  
(`hd0,0`)  $\Rightarrow$  `/dev/sda1`
    - Nombres de ficheros (`hd0,0`)/`boot/grub/grub.conf`



# Fragmento /boot/grub/grub.cfg (Linux)

```
1  ...
2  menuentry 'Debian GNU/Linux, con Linux 3.16.0-4-amd64' --class debian --class
   gnu-linux --class gnu --class os {
3      insmod part_msdos
4      insmod ext2
5      set root='hd0,msdos1'
6      search --no-floppy --fs-uuid --set=root bf8474c5-958e-4cca-a568-4828
        b2310fda
7      echo 'Cargando Linux 3.16.0-4-amd64...'
8      linux /boot/vmlinuz-3.16.0-4-amd64 root=UUID=bf8474c5-958e-4cca-a568
        -4828b2310fda ro initrd=/install/gtk/initrd.gz quiet
9      echo 'Cargando imagen de memoria inicial...'
10     initrd /boot/initrd.img-3.16.0-4-amd64
11
12 }
13 menuentry 'Debian GNU/Linux, with Linux 3.16.0-4-amd64 (recovery mode)' --class
   debian --class gnu-linux --class gnu --class os {
14     insmod part_msdos
15     insmod ext2
16     set root='hd0,msdos1'
17     search --no-floppy --fs-uuid --set=root bf8474c5-958e-4cca-a568-4828
        b2310fda
18     echo 'Cargando Linux 3.16.0-4-amd64...'
19     linux /boot/vmlinuz-3.16.0-4-amd64 root=UUID=bf8474c5-958e-4cca-a568
        -4828b2310fda ro single initrd=/install/gtk/initrd.gz
20     echo 'Cargando imagen de memoria inicial...'
21     initrd /boot/initrd.img-3.16.0-4-amd64
22 }
23 ...
```

# Fragmento /boot/grub/grub.cfg (Windows)

```
1  ...
2  ### BEGIN /etc/grub.d/30_os-prober ###
3  menuentry "Windows 7 (loader) (on /dev/sda1)" --class windows --class os {
4      insmod part_msdos
5      insmod ntfs
6      set root='(/dev/sda,msdos1)'
7      search --no-floppy --fs-uuid --set=root 28FCB3B0FCB376A2
8      chainloader +1
9  }
10 ### END /etc/grub.d/30_os-prober ###
11 ...
```



# Modo monousuario

```
GNU GRUB  version 1.99-27+deb7u1

Debian GNU/Linux, with Linux 3.2.0-4-686-pae
Debian GNU/Linux, with Linux 3.2.0-4-686-pae (recovery mode)

Use the + and - keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line.
```



# Modo monousuario

- Estado del sistema definido para realizar tareas **administrativas y de mantenimiento**, que requieren un control **completo** y **no compartido**.
- Sólo realiza el montaje del sistema de ficheros raíz (/), los otros SF están disponibles pero no están montados.
- Se puede acceder a todo el sistema, pero:
  - Muy pocos demonios están en ejecución, sólo los necesarios.
  - Muchas utilidades no están activas (impresión, red).
  - Sólo las órdenes del SF raíz están disponibles (si /usr está en otra partición, no está montado).
- Para entrar en modo monousuario el proceso **Init** crea el *shell* por defecto (/bin/sh) como usuario **root**:
  - Pero antes se ejecuta la orden /sbin/sulogin, que pide la contraseña de root para dejar entrar al sistema.





# Modo monousuario

- ¿Cómo se entra en modo monousuario?
  - Indicándolo manualmente al MBR con una opción o parámetro: mediante la interfaz de edición de GRUB, opción `single` a la entrada del núcleo.
  - Automáticamente, si hay problemas en el proceso de arranque que el sistema no puede solucionar por sí solo (p.e. problemas en el SF que `fsck` no puede solucionar, errores en los ficheros de arranque).
- **¡Problema!**: si cambiamos las opciones de GRUB y ponemos `init=/bin/sh`, no se llama a `sulogin`<sup>1</sup>.
  - Permite tener acceso a todo el sistema, estando delante del ordenador.

---

<sup>1</sup><https://blog.sleeplessbeastie.eu/2014/05/01/how-to-access-single-user-mode-without-password/>



# Modo monousuario

- **Solución:** no existe, salvo utilizar cifrado de ficheros.
- Al menos podemos paliarlo → solicitar contraseña para la entrada de administración.
  - Fichero /etc/grub.d/40\_custom (o donde esté la entrada).

```
1 set superusers="user1"  
2 password_pbkdf2 user1 grub.pbkdf2.sha512.10000.086EB0CC8 ...  
3 password_pbkdf2 user2 grub.pbkdf2.sha512.10000.045EB0CC8 ...
```

- Modificar la entrada de administración, para que requiera password, incluyendo --users user1 (modificarlo en los scripts).
- El password se puede generar usando:

```
1 pagutierrez@TOSHIBA:~$ grub-mkpasswd-pbkdf2  
2 Enter password:  
3 Reenter password:  
4 Your PBKDF2 is grub.pbkdf2.sha512.10000.086EB0CC8CB1E39E2...
```



# Modo monousuario

Vulnerabilidad en el sistema de contraseñas de grub:

- Enlace a la noticia (2016).



# Modo multiusuario

- Pasos del proceso de arranque (I/II):
  - ❶ Chequea el sistema de ficheros raíz con `fsck`.
    - Si al apagar el sistema, el sistema de ficheros se desmontó correctamente, no se chequea.
    - Sin embargo, algunos SOs con determinados SFs fuerzan el chequeo siempre, o cada cierto tiempo (cada 3 meses) o cada cierto número de montajes sin chequear (cada 20 veces).
    - Si `fsck` encuentra problemas que no puede solucionar “sólo”, lleva al sistema a modo monousuario para que el administrador realice el chequeo manual.
  - ❷ Monta el sistema de ficheros raíz en modo lectura-escritura.
  - ❸ Chequea el resto de SFs con `fsck` (idem al punto 1).
  - ❹ Monta el resto de SFs.
  - ❺ Activa las particiones de intercambio (*swapping*): `swapon -a`.
  - ❻ Activa las cuotas de disco: `quotacheck -a` y `quotaon -a`.



# Modo multiusuario

- Pasos del proceso de arranque (II/II):
  - 7 Lanza los procesos servidores o demonios:  
crond, atd, cupsd, syslogd...
  - 8 Activa la red.
  - 9 Lanza los demonios de red:  
xinetd, apache2, nagiosd, sshd, ntpd, nfsd, rpc.mountd, slapd...
  - 10 Limpia los sistemas de ficheros: /tmp, etc.
  - 11 Permite que los usuarios entren:
    - Crea las terminales, lanzando getty en modo texto, y el terminal gráfico, si es preciso.
    - Borra, en caso de que exista, el fichero /etc/nologin:  
Si el fichero /etc/nologin existe, los usuarios (excepto **root**) no pueden entrar al sistema. Algunos sistemas lo crean al iniciar el arranque.



# Niveles de ejecución en GNU/Linux

- El SO puede estar en distintos niveles de ejecución (no solo modo monousuario y multiusuario).
- En GNU/Linux, los niveles de ejecución son:
  - **Nivel 0:** Sistema apagado.
  - **Nivel 1, s o S:** Modo monousuario, rescue o troubleshooting.
  - **Nivel 2:** Modo multiusuario sin funciones de red.
  - **Nivel 3:** Modo multiusuario con funciones de red y terminales de texto.
  - **Nivel 4:** Sin usar, a redefinir por el administrador.
  - **Nivel 5:** Modo multiusuario con funciones de red e inicio de sesión gráfico.
  - **Nivel 6:** Sistema reiniciándose.
- En Debian, por defecto, los niveles 2 al 5 son todos modo multiusuario con todas las funciones.



# Niveles de ejecución en GNU/Linux

- `/sbin/runlevel`  $\Rightarrow$  saber en qué nivel está el sistema.
- `/sbin/telinit`  $\Rightarrow$  cambiar de nivel de ejecución:
  - `telinit 1`  $\rightarrow$  a modo monousuario.
  - `telinit 6`  $\rightarrow$  reiniciar el sistema.
  - `telinit 3`  $\rightarrow$  cambiar al nivel 3.
- El nivel por defecto, establecido al arrancar, se encuentra:
  - En el fichero `/etc/inittab`

```
id:2:initdefault:
```

- O en el fichero `/etc/init/rc-sysinit.conf` (upstart)

```
env DEFAULT_RUNLEVEL=2
```

- Al arrancar mediante GRUB, al núcleo se le puede pasar como parámetro un número indicando el nivel en el que queremos arrancar. En este caso se obviará el nivel por defecto.



# Ficheros de inicialización

- Personalizar niveles de ejecución  $\Rightarrow$  carpetas `/etc/rc?.d/`, donde `?` es el nivel de ejecución.
- Todos ellos son ejecutados por **Init** durante el arranque.
- Se ejecutan al arrancar o al cambiar de nivel:
  - El nombre del script empieza por S o K, seguido de dos dígitos y un nombre descriptivo:  
`K35smb K15httpd S40atd S50xinetd S60cups S99local`
  - Los ejecuta en orden alfabético, primero los **K** después los **S**, los dos dígitos establecen el orden entre todos los K y todos los S.
  - **Ficheros K**: detener demonios o matar procesos.
  - **Ficheros S**: lanzar demonios o ejecutar funciones de inicio.
  - Para cada nivel de inicialización, se especifica qué demonios tienen que estar activos o no.





# Ficheros de inicialización

- Carpetas `/etc/rc?.d/`:
  - Todos los ficheros son enlaces simbólicos al fichero con el mismo nombre descriptivo localizado en `/etc/init.d`.
  - Los *scripts* reciben varios parámetros: `start`, `stop`, `restart...`
  - Esto permite lanzar o relanzar demonios sin reiniciar el sistema.
  - `rc` ejecuta los ficheros `K` con el parámetro `stop` y los `S` con `start`.
  - Estos scripts están en desuso y se tiende a utilizar `upstart` y el comando `service`: muchos de los scripts simplemente llaman a `upstart`.



# Ficheros de inicialización

```
1 pedroa@pedroaLaptop:~$ ls /etc/rc2.d/ -la
2 total 20
3 drwxr-xr-x  2 root root  4096 oct 27 12:05 .
4 drwxr-xr-x 146 root root 12288 feb 21 16:11 ..
5 -rw-r--r--  1 root root   677 jul 14 2013 README
6 lrwxrwxrwx  1 root root   14 jul 25 2013 S01motd -> ../init.d/motd
7 lrwxrwxrwx  1 root root   17 jul 25 2013 S13rpcbind -> ../init.d/rpcbind
8 lrwxrwxrwx  1 root root   17 jul 25 2013 S16rsyslog -> ../init.d/rsyslog
9 lrwxrwxrwx  1 root root   14 jul 25 2013 S16sudo -> ../init.d/sudo
10 lrwxrwxrwx  1 root root   15 jul 25 2013 S17acpid -> ../init.d/acpid
11 lrwxrwxrwx  1 root root   17 jul 25 2013 S17anacron -> ../init.d/anacron
12 lrwxrwxrwx  1 root root   13 jul 25 2013 S17atd -> ../init.d/atd
13 lrwxrwxrwx  1 root root   14 jul 25 2013 S17cron -> ../init.d/cron
14 lrwxrwxrwx  1 root root   14 jul 25 2013 S17dbus -> ../init.d/dbus
15 lrwxrwxrwx  1 root root   15 jul 25 2013 S17exim4 -> ../init.d/exim4
16 lrwxrwxrwx  1 root root   17 jul 26 2013 S17hddtemp -> ../init.d/hddtemp
17 lrwxrwxrwx  1 root root   13 oct 27 12:05 S17ntp -> ../init.d/ntp
18 lrwxrwxrwx  1 root root   15 jul 25 2013 S17rsync -> ../init.d/rsync
19 lrwxrwxrwx  1 root root   13 jul 25 2013 S17ssh -> ../init.d/ssh
20 lrwxrwxrwx  1 root root   17 jul 27 2013 S19openvpn -> ../init.d/openvpn
21 lrwxrwxrwx  1 root root   14 jul 27 2013 S20cups -> ../init.d/cups
22 lrwxrwxrwx  1 root root   14 jul 27 2013 S20gdm3 -> ../init.d/gdm3
23 lrwxrwxrwx  1 root root   15 jul 27 2013 S20saned -> ../init.d/saned
```



# Manejar servicios

```
1 # Arrancar un servicio (tradicional):  
2 /etc/init.d/myservice start  
3 # Arrancar un servicio (upstart):  
4 service myservice start  
5 # Parar un servicio (tradicional):  
6 /etc/init.d/myservice stop  
7 # Parar un servicio (upstart):  
8 service myservice stop  
9 # Listar servicios (tradicional):  
10 ls /etc/init.d  
11 # Listar servicios (upstart):  
12 service --status-all  
13 # Añadir un servicio a todos los niveles  
14 update-rc.d apache2 defaults  
15 # Eliminar un servicio a todos los niveles  
16 rm /etc/rc*/myscript  
17 # Eliminar un servicio a todos los niveles  
18 update-rc.d apache2 remove
```



# Upstart

- `upstart`  $\Rightarrow$  proceso de arranque/parada del sistema basado en eventos, reemplazo del clásico **Init** (aunque los ficheros siguen denominándose `init`).
  - Convive con los Sysv *scripts*.
- Este proceso realiza, de forma asíncrona, las siguientes tareas:
  - Dirige el inicio de las tareas y demonios.
  - Controla los demonios mientras el sistema está encendido.
  - Detiene los demonios durante el proceso de apagado.
- En el directorio `/etc/init/` hay una serie de ficheros de configuración de eventos (`evento.conf`) que **Init** ejecuta según el orden y las dependencias establecidas en los mismos.
- Estos eventos indican qué tarea ejecutar, cuándo y cómo, mediante su propio lenguaje.



# Upstart

- `initctl`  $\Rightarrow$  permite al administrador interactuar con **Init**, para decirle que realiza determinadas acciones:

`start evento`                      `stop evento`                      `status evento`

- Ficheros de configuración de eventos (`.conf`):

- `exec <orden> <argumentos>`  $\Rightarrow$  ejecuta la orden con los argumentos indicados.

```
exec gdm-binary $CONFIG_FILE
```

```
exec acpid -c /etc/acpi/events -s  
/var/run/acpid.socket
```

- `script ... end script`  $\Rightarrow$  ejecutar el guión shell indicado:

```
1 script  
2     if [ -x /usr/share/recovery-mode/recovery-menu ]; then  
3         exec /usr/share/recovery-mode/recovery-menu  
4     else  
5         exec /sbin/sulogin  
6     fi  
7 end script
```



# Upstart

- Ficheros de configuración de eventos (.conf):
  - **start on <event>** ⇒ describe bajo qué condiciones se lanzará ese evento.

```
start on startup                start on runlevel 5
start on stopped rc2           start on started prefdm
```

- **stop on <event>** ⇒ describe bajo qué condiciones se parará ese evento.

```
stop on runlevel [35]          stop on started prefdm
```

- **respawn** ⇒ volver a lanzar ese proceso o demonio cuando se pare.
- **console** ⇒ hacia dónde redirigir la salida del evento.



# Upstart

- Ficheros de configuración de eventos (.conf):
  - **pre-start** ⇒ ejecutar la orden/guión shell antes de lanzar ese proceso:

```
1 pre-start exec rm -f /var/run/crond
2 pre-start script
3 if [ "$RUNLEVEL" == "S" ]
4 then
5     RUNLEVEL=1
6 fi
7 end-script
```

- **pre-stop** ⇒ ejecutar la orden/guión shell antes de parar ese proceso.



# Upstart

- Ficheros de configuración de eventos (.conf):
  - **post-start** ⇒ ejecutar la orden/guión shell después de lanzar ese proceso:

```
1 post-start exec touch /var/run/crond
2 post-start script
3     if [ "$RUNLEVEL" == "1" ]
4     then
5         RUNLEVEL=S
6     fi
7 end-script
```

- **post-stop** ⇒ ejecutar la orden/guión shell después de parar ese proceso.





# Upstart: /etc/init/rc-sysinit.conf |

```
1 # rc-sysinit - System V initialisation compatibility
2 #
3 # This task runs the old System V-style system initialisation scripts,
4 # and enters the default runlevel when finished.
5 description "System V initialisation compatibility"
6 author "Scott James Remnant <scott@netsplit.com>"
7
8 start on filesystem and net-device-up IFACE=lo
9 stop on runlevel
10
11 # Default runlevel, this may be overridden on the kernel command-line
12 # or by faking an old /etc/inittab entry
13 env DEFAULT_RUNLEVEL=2
14
15 emits runlevel #Evento que se crea
16
17 # There can be no previous runlevel here, but there might be old
18 # information in /var/run/utmp that we pick up, and we don't want that.
19 #
20 # These override that
21 env RUNLEVEL=
22 env PREVLEVEL=
23 console output      #Salida estándar a la consola
24 env INIT_VERBOSE
25
26 task
27
28 script
29     # Check for default runlevel in /etc/inittab
30     if [ -r /etc/inittab ]
31     then
```

# Upstart: /etc/init/rc-sysinit.conf II

```
32         eval "$(sed -nre 's/^[^#][^:]*:([0-6sS]):initdefault:./DEFAULT_RUNLEVEL
33         ="1";/p' /etc/inittab || true)"
34     fi
35     # Check kernel command-line for typical arguments
36     for ARG in $(cat /proc/cmdline)
37     do
38         case "${ARG}" in
39             -b|emergency)
40                 # Emergency shell
41                 [ -n "${FROM_SINGLE_USER_MODE}" ] || sulogin
42                 ;;
43             [0123456sS])
44                 # Override runlevel
45                 DEFAULT_RUNLEVEL="${ARG}"
46                 ;;
47             -s|single)
48                 # Single user mode
49                 [ -n "${FROM_SINGLE_USER_MODE}" ] || DEFAULT_RUNLEVEL=S
50                 ;;
51             esac
52     done
53
54     # Run the system initialisation scripts
55     [ -n "${FROM_SINGLE_USER_MODE}" ] || /etc/init.d/rcS
56
57     # Switch into the default runlevel
58     telinit "${DEFAULT_RUNLEVEL}"
59 end script
```

# Resumen del proceso de arranque

- Iniciador ROM:
  - Chequeo inicial del sistema.
  - Lee y almacena en memoria el programa cargador del SO.
  - Pasa el control al cargador del SO, saltando a la dirección de memoria donde lo ha almacenado.
- Cargador del sistema operativo (GRUB)  $\Rightarrow$  carga el núcleo del SO y le pasa el control, sabe dónde está el núcleo.
- Núcleo del SO:
  - Chequeo *hardware*.
  - Creación e inicialización de las estructuras de datos, tablas...
  - Crea el proceso **Init** y le pasa el control.
- Proceso **Init**: termina el proceso de arranque, dejando el sistema preparado para ser usado (chequeo de SFs, montaje de SFs, activación de la swap, de cuotas, demonios, etc.)



# Parada del sistema

- En ocasiones es necesario apagar o reiniciar el sistema: mantenimiento, diagnóstico, hardware nuevo, etc.

## Acciones durante proceso de parada

- 1 Se notifica a los usuarios.
- 2 Procesos en ejecución  $\Rightarrow$  enviar la señal de terminación (**TERM**).
- 3 Se paran los demonios.
- 4 A los usuarios que quedan conectados se les echa del sistema.
- 5 Procesos que queden en ejecución  $\Rightarrow$  enviar la señal de fin (**KILL**).
- 6 Actualizaciones de disco pendientes (integridad del SF) con **sync**.



# Parada del sistema: shutdown

- **shutdown [opciones] tiempo [mensaje]:**
  - Sin opciones: modo monousuario (telinit 1).
  - -r: reiniciar (telinit 6).
  - -h: parar (telinit 0).
  - -c: cancelar.
  - -k: hacer una simulación de apagado.
  - tiempo: +minutos, now, horas:minutos.
- Al salir del modo monousuario, vuelve al nivel por defecto (salvo que expresamente se reinicie o apague).



# Caídas del sistema y problemas de arranque

## Posibles causas de caídas del sistema

- Fallos *hardware*.
- Errores de *hardware* irrecuperables.
- Fallos de luz (cortes o altibajos).
- Otros problemas ambientales.
- Problemas de entrada/salida.
- Problemas de algún sistema de ficheros.



# Caídas del sistema y problemas de arranque

## Problemas de arranque

- Fallos *hardware*.
- No se puede leer el sistema de ficheros de los discos de trabajo.
- Hay áreas dañadas en el disco que no pertenecen al sistema de ficheros (p.e. tabla de particiones).
- *Hardware* incompatible.
- Errores en la configuración del sistema.



# Caídas del sistema y problemas de arranque

- Al rearrancar mirar los mensajes que hay en el fichero `/var/log/messages`.
- La orden `dmesg`  $\Rightarrow$  mensajes producidos durante el arranque.
- En el arranque al núcleo se le pueden pasar otros parámetros:
  - `root=particion`  $\Rightarrow$  indicar que monte como partición raíz una distinta.
  - `init=ejecutable`  $\Rightarrow$  que en vez del proceso **Init** lance otro proceso:  
`init=/bin/bash`  $\Rightarrow$  en este caso el proceso de inicio del **Init** no se realiza, el SF está montado en modo sólo lectura, hay que remontarlo: `mount -o remount -w -n /`
  - `single`  $\Rightarrow$  arrancar en modo monousuario.
  - Un número indicando el nivel de arranque.





# Referencias



Evi Nemeth, Garth Snyder, Trent R. Hein y Ben Whaley

Unix and Linux system administration handbook.

Capítulo 3. *Booting and shutting down*, Capítulo 11. *System and log files*.

Prentice Hall. Cuarta edición. 2010.



Aeleen Frisch.

Essential system administration.

Capítulo 4. *Startup and shut down*.

O'Reilly and Associates. Tercera edición. 2002.



# Programación y Administración de Sistemas

## 3. Arranque y parada del sistema

Pedro Antonio Gutiérrez

Asignatura "Programación y Administración de Sistemas"  
2º Curso Grado en Ingeniería Informática  
Escuela Politécnica Superior  
(Universidad de Córdoba)  
[pagutierrez@uco.es](mailto:pagutierrez@uco.es)

22 de febrero de 2017

