




Capítulo 3

INICIO DEL SISTEMA Y PROCESO DE LOGIN

La secuencia de arranque

- BIOS o UEFI realizan una verificación inicial del hardware del equipo.
  - Esta revisión recibe el nombre de POST (Power-On Self Test).
- Se busca un gestor de arranque (por lo general, **GRUB**) en el **MBR** (**M**aster **B**oot **R**ecord) o en la partición **EFI**
- El control del proceso se pasa a manos de **GRUB**
  - **GRUB** carga el kernel, el cual reconocerá y configurará los dispositivos de hardware presentes en el equipo y los preparará para su uso.
- El núcleo también será responsable por ejecutar el primer proceso, también conocido como **init**.
  - A continuación, **init** utilizará el gestor del sistema (**systemd**, **Upstart**, o los scripts clásicos de **SysVinit**) para continuar el inicio.
- El proceso de arranque culmina al presentar una interfaz de inicio de sesión de modo texto o gráfico, según sea el caso.
- *Los servicios o procesos son programas que corren en segundo plano (de manera no interactiva). Otro nombre por el que son conocidos es **daemons** (a veces traducido como **demonios** en castellano).*

SysVinit

- Hasta no hace mucho tiempo, la mayoría de las distribuciones más utilizadas utilizaban como base de su funcionamiento el sistema de arranque y de administración de servicios conocido como **SystemV** o **SysVinit**.
  - Heredado de Unix
  - Contempla 5 niveles útiles de funcionamiento (de ahí el nombre *System**V***, correspondiente al número 5 en el sistema romano) numerados del 1 al 5. A estos se les suma el nivel 0 (apagado del equipo) y el 6 (reinicio del sistema). Cada uno de ellos es lo que se conoce como *niveles de corrida* o *runlevels* en Linux.
  - Cada *runlevel* se encuentra asociado a un cierto número de servicios que por defecto deben iniciarse automáticamente cuando encendemos el equipo, y que deben detenerse al reiniciarlo o apagarlo.
- Dentro de **/etc** encontramos 7 directorios con el nombre **rc*N*.d**, donde ***N*** es un número del 0 al 6.
  - Es dentro de estos directorios que se encuentran una serie de enlaces simbólicos a los ejecutables que inician y detienen los servicios del sistema en cada *runlevel* correspondiente.
  - Podemos considerar (por el momento) a los enlaces simbólicos como los accesos directos disponibles en otros sistemas operativos.

SysVinit (cont.)

- Lo primero que hace **init** es leer el archivo **/etc/inittab** para identificar los próximos pasos a seguir.
- En este archivo se indica el nivel de corrida por defecto:

```
# The default runlevel.
id:2:initdefault:
```

- A continuación, **init** se dirige a **/etc/rc2.d** en este caso. En este directorio podemos apreciar una serie de enlaces simbólicos cuyos nombres comienzan con la letra **S**, tal como mencionamos antes, y un número de dos dígitos. Cada uno de ellos apunta a un servicio que se debe levantar en el *runlevel* 2 por orden numérico y alfabético).
- Limitaciones de SysVinit:
  - Estrictamente síncrono
  - Chequear dependencias
  - Eventos posteriores al inicio requieren intervención del usuario

Upstart

- Desarrollado por Canonical e integrada por primera vez en Ubuntu 6.10 Edgy
  - Fedora la adoptó posteriormente y la utilizó hasta la versión 14
  - Seguida por RHEL 6 (y CentOS 6) hasta las versiones 6.7, con soporte hasta fines de 2018
- Reemplazo para **SysVinit** basado en eventos
- Responde a conexión y desconexión de dispositivos extraíbles
- Es 100% compatible con los scripts clásicos de **SysVinit**
- También trabaja con archivos de configuración propios (.conf) dentro del directorio /etc/init
  - Estructura:
    - Descripción del proceso
    - Niveles de corrida en los cuales debe ejecutarse o eventos que deben iniciarlo
    - Niveles de corrida en los cuales no debe correr o eventos que deben deternerlo
    - Opciones
    - Comando a utilizar para iniciar el proceso

Upstart (cont.)

- Por ejemplo:

```
# Servicio de prueba para Upstart
# Stanzas
# Stanzas define when and how a process is started and stopped
# See a list of stanzas here: http://upstart.ubuntu.com/wiki/Stanzas#respawn

# Runlevels para iniciar el servicio
start on runlevel [2345]
# Cuando detenerlo
stop on runlevel [016]
# Reiniciarlo en caso de falla
respawn
# Especificar directorio de trabajo
chdir /home/alumno/misarchivos
# Indicar el comando, junto con cualquier argumento necesario (opcional)
exec bash backup.sh arg1 arg2
```

Systemd

- No surgió como reemplazo a **SysVinit** porque este ultimo fuera defectuoso, ni a causa del descontento de los administradores
- La idea era desarrollar un sistema más eficiente que:
  - Iniciara menos servicios durante el arranque (solamente aquellos que fueran necesarios de acuerdo al uso esperado y al hardware disponible).
  - Lo hiciera en paralelo, en vez de manera secuencial.
- En otras palabras, se buscó un sistema de inicio y de administración de servicios que pudiera reaccionar *dinámicamente* ante cambios en el software y en el hardware.
  - Por ejemplo, no hay necesidad de mantener el servicio de impresión corriendo si no hay ninguna impresora conectada al equipo o disponible a través de la red. Por otra parte, queremos que se inicie de manera transparente si esa condición cambia.
- Muchas personas y parte de la comunidad se opuso a **systemd**. Sin embargo, eventualmente fue adoptado en Fedora (luego en RHEL y CentOS, naturalmente) y más adelante en ArchLinux, Debian, Ubuntu, y derivados.

Targets en systemd

- El componente básico de **systemd** se denomina *unidad* o *unit*.
- Existen varias categorías de unidades. Las más conocidas son
  - Los *servicios* (*services*) ejecutan los *daemons* y sus dependencias en el orden apropiado.
  - Los *objetivos* (*targets*) agrupan dos o más unidades.
- Los *targets* u *objetivos* en **systemd** son análogos a los niveles de corrida de **SysVinit**
- En **/lib/systemd/system** encontramos los archivos de definición de las unidades
- En particular, los correspondientes a los objetivos tienen la extensión **.target**.
- Por ejemplo:
  - **basic.target** incluye los servicios indispensables del sistema
  - **multi-user.target** agrupa la mayoría de los *daemons*
  - **basic.target** es una dependencia de **multi-user.target**.

Target por defecto

- Para conocer el *target* por defecto: `systemctl get-default`
- Se trata de un enlace simbólico llamado **default.target** dentro de **/lib/systemd/system**
  - Apunta a la definición del *target* indicado.
- Por ejemplo, en la imagen podemos ver que **/lib/systemd/system/default.target** es un *soft link* hacia **/lib/systemd/system/graphical.target**
- Es posible cambiar el *target* por defecto (más adelante en el curso veremos cómo hacerlo)

```
root@stretch:~# ls -l /lib/systemd/system/default.target
lrwxrwxrwx 1 root root 16 jul  5  2017 /lib/systemd/system/default.target -> graphical.target
root@stretch:~# cat /lib/systemd/system/default.target
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.service
AllowIsolate=yes
root@stretch:~#
```

Definición del objetivo por defecto

Podemos decir lo siguiente en general:
Si en **A.target** se indica **Requires=B.target**, hablamos que el **objetivo B** es una **dependencia del A**.

El proceso de login

1) Al ingresar usuario y contraseña, se compara contra **/etc/passwd** y **/etc/shadow**.
   - Info de usuario y contraseñas hasheadas con SHA-512.
2) Si las credenciales son correctas, se inicia la sesión y se nos muestra el contenido de **/etc/motd**.
   - Este archivo ("**M**essage **O**f **T**he **D**ay") puede ser utilizado por los administradores para dejar un mensaje a los usuarios.
3) El sistema operativo nos devuelve el *prompt*, es decir la línea de comandos lista para aceptar nuestras órdenes.
4) En Debian, en el *prompt* se muestran:
   - Nombre del usuario actual
   - Nombre del equipo
   - Directorio actual de trabajo

El proceso de login (cont.)

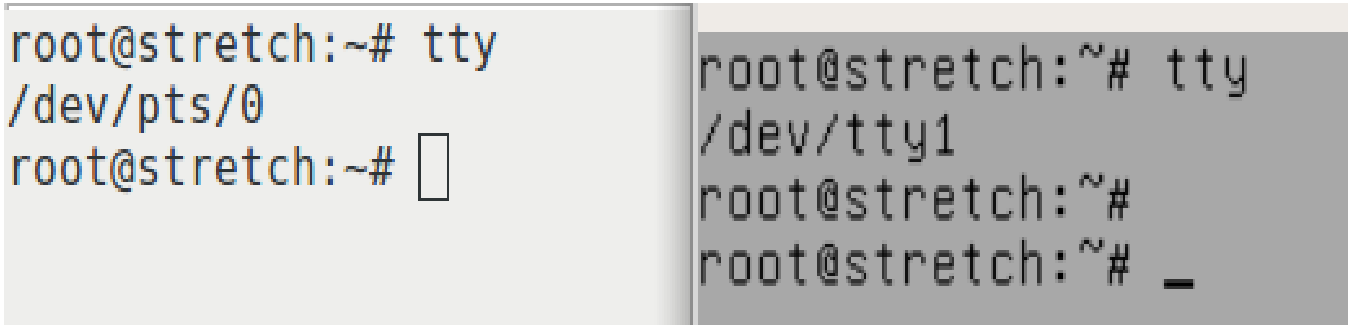
- Además de su directorio personal, cada usuario tiene asignado un *shell* o *intérprete de comandos*. Se trata de un programa que *recibe* los comandos que escribimos y que los *envía* al sistema operativo para ser ejecutados.
- El *intérprete* utilizado en la mayoría de las distribuciones actuales (de las cuales Debian no es la excepción) se llama **Bash** (**B**ourne-**A**gain **SH**ell). **Bash** tiene la posibilidad de ejecutar comandos en tiempo real pero además tiene un poderoso lenguaje de programación de scripts. Permite generar programas con funciones, control de flujo, creación de archivos, seguimiento de procesos, entre otros.

El modo texto

- El modo texto es la interfaz más interesante para comenzar nuestro aprendizaje de Linux.
  - Nos permite interactuar con el sistema operativo desde el principio y muy fácilmente.
- Una consola es una terminal física conectada directamente a una máquina.
- Una *terminal* (también llamada **tty**) es un recurso del sistema donde se pueden escribir comandos y ver el resultado de los mismos.
  - Hoy en día asociamos las terminales con tty1, tty2, hasta tty6, que están disponibles para ser utilizadas por distintos usuarios al presionar la combinación de teclas Ctrl+Alt+F1, Ctrl+Alt+F2, y así sucesivamente (¡recordemos que Linux es un sistema operativo multiusuario y multitarea!).
- Si en vez de trabajar en el modo texto exclusivamente también hemos instalado una distribución con un entorno gráfico, dispondremos de una aplicación llamada **Terminal**. Este programa nos permite acceder a la línea de comandos mediante una interfaz gráfica, por lo que recibe el nombre de *pseudo terminal* o **pts**.

El modo texto (cont.)

- Utilizamos el comando `tty` para observar la terminal que estamos utilizando actualmente.
- La figura siguiente muestra dos sesiones abiertas del usuario root. Una de ellas está utilizando la **tty1** y la otra **pts0**.



Los primeros comandos

- Tres comandos esenciales para navegar el sistema:
  - `pwd` para mostrar el nombre del directorio actual de trabajo (el lugar dentro de la estructura de directorios donde estemos posicionados en un momento dado).
  - `ls` para mostrar los contenidos de un directorio.
  - `cd` para cambiar de directorio.
- Ejecutemos los dos primeros y veamos el resultado:

```
alumno@stretch:~$ pwd
/home/alumno
alumno@stretch:~$ ls
adm.txt                archivo2           dato
alumnosconectados.txt  archivo3           ejem
archivo1               claseloperador.txt  for-
alumno@stretch:~$
```

Recordemos del capítulo 2 que **alumno** es un subdirectorio de **/home**, que a su vez se encuentra dentro de **/**. Para verificarlo, podemos cambiar de directorio a **/** y al listar sus contenidos, veremos allí al directorio **home**.

```
alumno@stretch:~$ cd /
alumno@stretch:/$ ls
archivos  boot     curso  etc       general  ini
bin       comunes  dev    finanzas  home     ini
alumno@stretch:/$ cd home
alumno@stretch:/home$ ls
alumno  fulano  gabriel  lost+found  pruebasamba
alumno@stretch:/home$
```

Los primeros comandos (cont.)

- Para cambiar de directorio utilizamos el comando `cd` seguido de la ubicación a la que deseamos movernos.
  - Lo mismo aplica a `ls`, que no solamente nos permite ver los contenidos de la ubicación actual sino también la de cualquier directorio del sistema (si es que contamos con los permisos para poder acceder al mismo, tema que trataremos más adelante).
- Sin importar el directorio en el que estemos parados, podemos utilizar `cd` *a secas* para regresar a nuestro *home* o `cd -` para dirigirnos al último directorio en el que estuvimos.
- En todos los comandos que requieren el uso de una ubicación dentro de la estructura de directorios, esta puede especificarse de dos maneras:
  - *Ruta absoluta* es la que comienza en **/** y va hasta el directorio o archivo en cuestión. Por ejemplo, para hacer referencia al archivo **adm.txt** dentro del *home* del usuario **alumno** utilizamos **/home/alumno/adm.txt**.
  - *Ruta relativa* es la que parte de la ubicación actual. Por ejemplo, si estamos parados en **/home** podemos movernos al *home* de alumno con el comando `cd alumno` en vez de `cd /home/alumno` (que sería el equivalente utilizando la ruta absoluta).