

## 2. Agile SysAdmin: Networking and Systems Administration

Instructor: Julián García-Sotoca Pascual





# ■ Usando la Interfaz de Línea de Comandos



# ■ Índice

- Linux
  - Comandos básicos
  - Usuarios y grupos
  - Herramientas
  - Instalación de Software
  - Instalación automatizada y arranque
  - Kernel: módulos y tuning



# ■ Índice

- Linux
  - **Comandos básicos**
  - Usuarios y grupos
  - Herramientas
  - Instalación de Software
  - Instalación automatizada y arranque
  - Kernel: módulos y tuning



# ■ Usando la CLI - Comandos básicos

## Fundamentos:

- Estándar POSIX (Portable Operating System Interface uniX)
- Unix Philosophy:
  - Hacer que cada programa haga una cosa y la haga bien
  - Hacer que los programas puedan trabajar juntos
  - Hacer que los programas puedan manejar inputs de texto, porque es una interfaz universal
- Con parámetros para modificar el comportamiento de los comandos

[http://emulator.pdp-11.org.ru/misc/1978.07\\_-\\_Bell\\_System\\_Technical\\_Journal.pdf](http://emulator.pdp-11.org.ru/misc/1978.07_-_Bell_System_Technical_Journal.pdf)



# ■ Usando la CLI - Comandos básicos

Fundamentos:

- Shell → intérprete de comandos. El más común es BASH
- Comandos pueden ser del sistema o de la Shell (Builtin Commands)
- Buscando ayuda:
  - --help o -h
  - manpages
  - info
  - Google.



# ■ Usando la CLI - Comandos básicos

Diferencia entre Consola, Terminal y Shell:

- Terminal: dispositivo para enviar comandos a una computadora y mostrar su respuesta. En el mundo software corresponde con el programa que emula este dispositivo.
- Consola: dispositivo con monitor y teclado. que ejecuta un terminal por software. Actualmente Terminal y Consola son sinónimos
- Shell: Es el programa donde el terminal envía los comandos



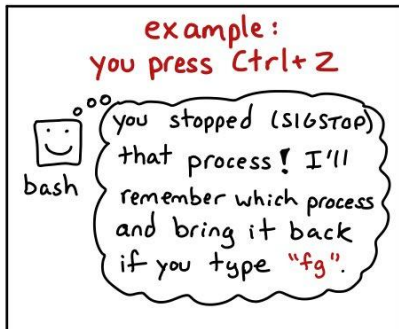
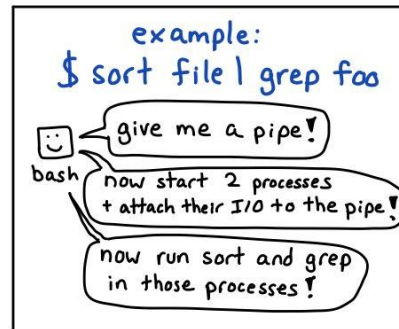
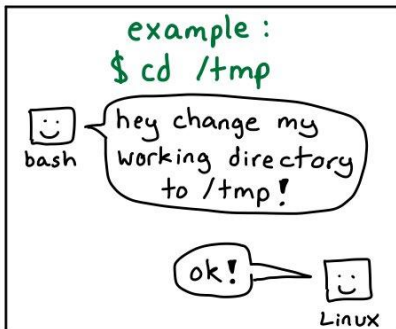
[Whats The Difference Between A Console A Terminal And A Shell](#)



# Usando la CLI - Comandos básicos

JULIA EVANS  
@b0rk

## What's a shell?





# ■ Usando la CLI - Comandos básicos

**Moviéndonos** por el sistema de ficheros:

- Listar ficheros → *ls*
- Cambiar de directorio → *cd*
- Saber en qué directorio estamos → *pwd*
- Crear enlaces → *ln*
- Crear ficheros o modificar hora de acceso → *touch*
- Crear un directorio → *mkdir*
- Cambiar permisos a un fichero → *chmod*
- Cambiar usuario a un fichero → *chown*
- Cambiar el grupo de un fichero → *chgrp*



# ■ Usando la CLI - Comandos básicos

**Moviéndonos** por el sistema de ficheros:

- Borrar ficheros → *rm*
- Borrar directorios → *rmdir*
- Copiar ficheros → *cp*
- Mover ficheros → *mv*
- Determinar el tipo de fichero → *file*
- Búsqueda de archivos → *find*



# ■ Usando la CLI - Comandos básicos

**Moviéndonos** por el sistema de ficheros:

- Se puede acceder por el camino absoluto o relativo
- / origen o raíz del sistema de ficheros → root
- “.” se refiere al directorio actual
- “..” se refiere al directorio superior
- “~” se refiere al home del usuario
- Caracteres comodín → “\*”, “?”, “[ ]”, “{ }”, “!”, “\”



# ■ Usando la CLI - Comandos básicos

- **Manejando ficheros**

- Con el comando *stat* obtenemos información de ficheros
  - Tamaño
  - Tipo
  - Permisos
  - Inodo

```
vagrant@ubuntu1:~$ stat fichero1.txt
  File: fichero1.txt
  Size: 0             Blocks: 0          IO Block: 4096   regular empty file
Device: 801h/2049d   Inode: 62628        Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/  vagrant)   Gid: ( 1000/  vagrant)
Access: 2245-10-18 09:53:01.131122049 +0000
Modify: 2245-10-18 09:53:01.131122049 +0000
Change: 2245-10-18 09:53:01.131122049 +0000
 Birth: -
```



# ■ Usando la CLI - Comandos básicos

- Manejando ficheros
  - ¿que es un **inodo**?
    - Todo sistema de ficheros mantiene una información de todos los objetos que almacena (metadata)
    - El inodo guarda la información de esos objetos:
      - dónde están localizados en el disco
      - tiempo de modificación
      - parámetros de seguridad
      - id de inodo
    - Todo sistema de ficheros se crea con un número finito de inodos, calculado en función de su tamaño



# ■ Usando la CLI - Comandos básicos

- Manejando ficheros

- **Enlaces**

- Duro (hard) o Simbólico (soft)
    - Se crean con el comando *ln*  
Hard link → *ln ORIGEN DESTINO*  
Soft link → *ln -s ORIGEN DESTINO*
    - Equivalente al “Acceso directo” de sistemas Windows
    - El enlace duro comparte el inodo → **es el mismo fichero físico!!**
    - Útiles para no crear múltiples copias de un fichero

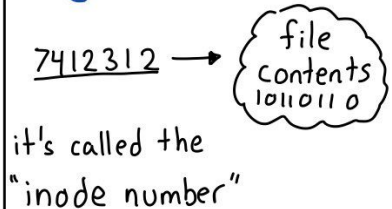


# Usando la CLI - Comandos básicos

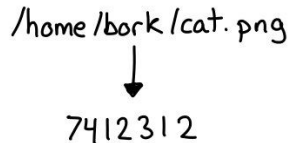
JULIA EVANS  
@b0rk

## inodes + hard links

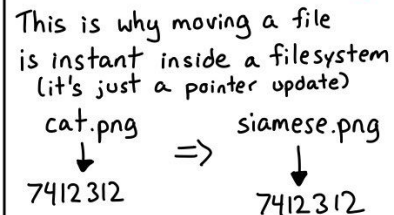
in a filesystem,  
every file has a number



a file name points to  
an inode number



move/edit a file:  
inode number won't change



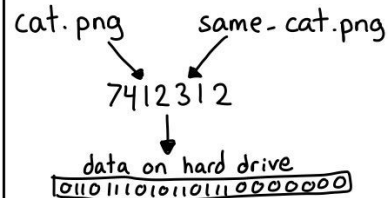
ls -li shows you  
inode numbers

```
$ ls -li
7412312 cat.png
7226709 dog.png
  ↑
the inode number!
```

"hard links" add a  
pointer to the  
same inode

```
$ ln cat.png same_cat.png
$ ls -li
7412312 cat.png
7412312 same_cat.png
  ↑
same inode!
```

hard links don't use  
extra disk space



# ■ Usando la CLI - Comandos básicos

- Manejando ficheros
  - Características de los **enlaces simbólicos**
    - Pueden apuntar a otro filesystem, local o remoto, y también a directorios
    - En la salida del ls se identifican con una l en la primera columna
    - Puede apuntar a un fichero que no existe → enlace roto
    - Utiliza los permisos del fichero origen





# ■ Usando la CLI - Comandos básicos

- Manejando ficheros
  - Características de los **enlaces duros**:
    - Al compartir inodo tienen diferente ruta absoluta pero comparten datos, propietario y permisos
    - Cuando se borra un fichero únicamente se borra el metadato de la ruta.
    - Únicamente se borra el inodo cuando el contador de inodos queda en 0
    - Deben residir en el mismo Filesystem
    - No pueden apuntar a directorios



# ■ Usando la CLI - Comandos básicos

## Procesar cadenas de texto:

- Mostrar el contenido de todo un archivo → *cat* o *tac*
- Desplazarnos dentro de un fichero → *more* o *less*
- Mostrar el inicio de un archivo → *head*
- Mostrar el final de un archivo → *tail*
- Contar líneas, palabras o caracteres → *wc*
- Sustituir tabulaciones por espacios simples → *expand*
- Sustituir espacios simples por tabulaciones → *unexpand*
- Mostrar archivos binarios → *hexdump*
- Convertir entre diferentes formatos de datos → *od*
- Buscar cadenas de caracteres en un fichero binario → *strings*



# ■ Usando la CLI - Comandos básicos

## Procesar cadenas de texto:

- Dividir un archivo en otros menores → *split*
- Mostrar el contenido eliminando líneas secuenciales repetidas → *uniq*
- Delimitar un archivo en columnas → *cut*
- Concatena archivos en columnas → *paste* o *join*
- Ordenar un fichero → *sort*
- Formatear un texto con un número de caracteres por línea → *fmt*
- Convertir caracteres → *tr*
- Comparar dos ficheros de texto → *diff*



# ■ Usando la CLI - Comandos básicos

## **Buscar** en archivos de texto:

- Búsqueda de un texto en el contenido de un fichero → *grep*
- Permite el uso de expresiones regulares
- opciones más comunes:
  - i: ignora la diferencia de mayúsculas y minúsculas
  - C n: muestra las n líneas anteriores y posteriores
  - v: muestra todas las líneas excepto la que corresponde a la expresión
  - r: busca recursivamente en una ruta
- Variaciones:
  - egrep: permite el uso de expresiones regulares extendidas y del operador “|” que actúa como un *OR*
  - fgrep: no interpreta expresiones regulares



# ■ Usando la CLI - Comandos básicos

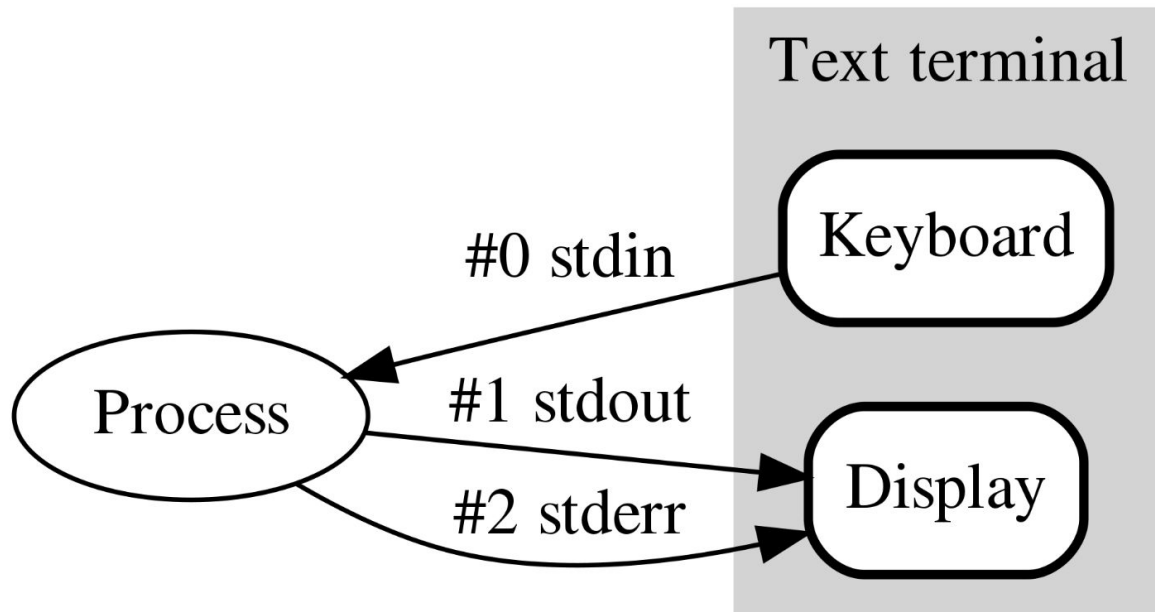
## Flujos, pipes y redireccionamiento:

- Los procesos en Unix abren 3 descriptores
  - stdin → standard input. Valor numérico **0**
  - stdout → standard output. Valor numérico **1**
  - stderr → standard error. Valor numérico **2**
- Dispositivos virtuales: /dev/stdin, /dev/stdout y /dev/stderr
- stdin normalmente hace referencia al teclado
- stdout y stderr normalmente hace referencia a la pantalla



# ■ Usando la CLI - Comandos básicos

Flujos, pipes y redireccionamiento:



[https://en.wikipedia.org/wiki/Standard\\_streams](https://en.wikipedia.org/wiki/Standard_streams)




# Tuberías y redirecciones

SULIA EVANS  
@b0rk

## file descriptors

Unix systems use integers to track open files



these integers are called **file descriptors**

**ls**of (list open files) will show you a process's open files

```
$ ls -p 4242
```

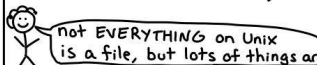
← PID we're interested in

FD	NAME
0	/dev/pts/tty1
1	/dev/pts/tty1
2	pipe:29174
3	/home/bork/awesome.txt
5	/tmp/

↑  
FD is for file descriptor


file descriptors can refer to:

- files on disk
- pipes
- sockets (network connections)
- terminals (like xterm)
- devices (your speaker! /dev/null!)
- LOTS MORE (eventfd, inotify, signalfd, epoll, etc etc)



not EVERYTHING on Unix is a file, but lots of things are

When you read or write to a file/pipe/network connection you do that using a file descriptor



connect to google.com

Write GET / HTTP/1.1 to fd #5


OS: ok! fd is 5! done!

Let's see how some simple Python code works under the hood:

Python:

```
f = open("file.txt")  
f.read lines()
```

Behind the scenes:



Python program: open file.txt, read from file #4

OS: ok! fd is 4, here are the contents!

(almost) every process has 3 standard FDs

```
stdin → 0  
stdout → 1  
stderr → 2
```

"read from stdin" means "read from the file descriptor 0"

could be a pipe or file or terminal

File descriptors



# ■ Usando la CLI - Comandos básicos

Flujos, pipes y redireccionamiento:

- Para redireccionar la salida estándar a un archivo → “>”
- Si no se quiere sobrescribir el archivo → “>>”
- Para redireccionar el contenido de un archivo a la entrada estándar → “<”
- Para especificar la salida:
  - “2>” → solo redirige el stderr al fichero
  - “&>” → redirige el stdout y stderr al fichero





# ■ Usando la CLI - Comandos básicos

Flujos, pipes y redireccionamiento:

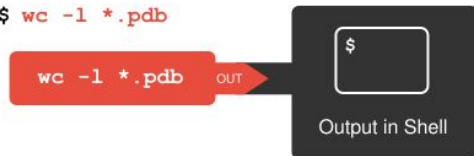
- La tubería (pipe) permite enviar la salida de un comando a la entrada de otro. Se especifica con el carácter “|”
- Con el comando *tee* se puede redireccionar simultáneamente a un archivo y a la stdout
- Se puede utilizar la salida de un comando como argumento para otro con comillas invertidas “`”



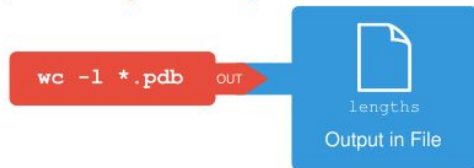
# Usando la CLI - Comandos básicos

Flujos, pipes y redireccionamiento:

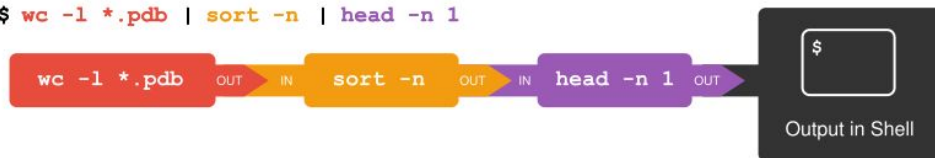
```
$ wc -l *.pdb
```



```
$ wc -l *.pdb > lengths
```



```
$ wc -l *.pdb | sort -n | head -n 1
```



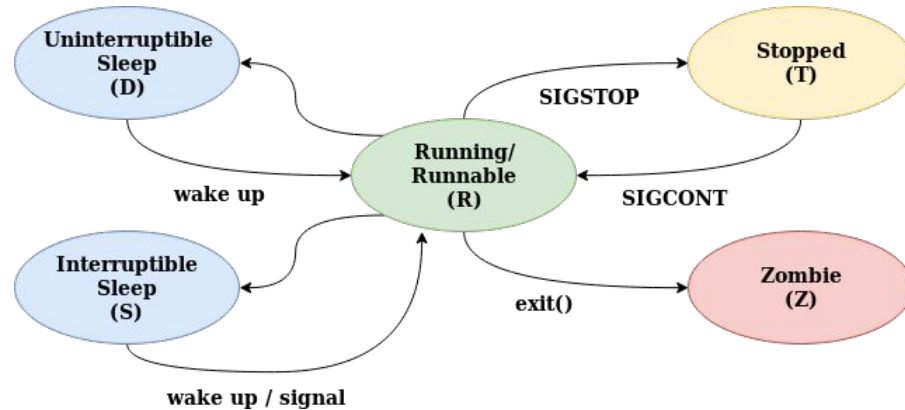
<https://swcarpentry.github.io/shell-novice/04-pipeline/index.html>



# ■ Usando la CLI - Comandos básicos

Procesos:

- Un proceso es un programa en ejecución
- Cada proceso está identificado por un PID
- Linux es multitarea permitiendo arrancar más de un proceso simultáneamente.
- Estados de un proceso:



[https://medium.com/@shereef\\_pt/process-in-a-linux-based-system-5f6434e8243](https://medium.com/@shereef_pt/process-in-a-linux-based-system-5f6434e8243)

<https://medium.com/@cloudchef/linux-process-states-and-signals-a967d18fab64>



# ■ Usando la CLI - Comandos básicos

Posibles estados de Procesos:

## PROCESS STATE CODES

Here are the different values that the `s`, `stat` and `state` output specifiers (header "STAT" or "S") will display to describe the state of a process:

D	uninterruptible sleep (usually IO)
I	Idle kernel thread
R	running or runnable (on run queue)
S	interruptible sleep (waiting for an event to complete)
T	stopped by job control signal
t	stopped by debugger during the tracing
W	paging (not valid since the 2.6.xx kernel)
X	dead (should never be seen)
Z	defunct ("zombie") process, terminated but not reaped by its parent

(man ps)



# ■ Usando la CLI - Comandos básicos

Procesos:

- Monitorizar procesos:
  - *ps*: muestra procesos activos
  - *top* o *htop*: monitorización continua
  - *pstree*: muestra el árbol de procesos
  - *pidof*: devuelve el PID de un programa
  - *kill*: envía señales de control para procesos
  - *killall* o *pkill*: utiliza el nombre del proceso en lugar del PID
  - *nice* o *renice*: nos permite gestionar las prioridades de un proceso



# Usando la CLI - Comandos básicos

Procesos:

- Monitorizar procesos:

```
1 [ 0.0% Tasks: 28, 16 thr; 1 running
2 [ 0.0% Load average: 0.03 0.03 0.02
Mem[|||||] 46.9M/488M Uptime: 00:10:30
Swp[ 0K/0K]

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
2391 vagrant 20 0 25496 4848 3280 R 1.3 1.0 0:00.31 htop
1134 root 20 0 268M 6176 5480 S 0.0 1.2 0:00.05 /usr/lib/accountsservice/accounts-daemon
2107 vagrant 20 0 93104 5028 3968 S 0.0 1.0 0:00.15 sshd: vagrant@pts/0
1116 root 10 -10 5720 3512 2432 S 0.0 0.7 0:00.22 /sbin/iscsid
1244 root 20 0 241M 2704 2324 S 0.0 0.5 0:00.09 /usr/sbin/VBoxService
1240 root 20 0 241M 2704 2324 S 0.0 0.5 0:00.16 /usr/sbin/VBoxService
1164 messagebu 20 0 42896 3988 3504 S 0.0 0.8 0:00.11 /usr/bin/dbus-daemon --system --address=systemd: --nof
1235 root 20 0 241M 2704 2324 S 0.0 0.5 0:00.30 /usr/sbin/VBoxService
1 root 20 0 117M 5896 3944 S 0.0 1.2 0:04.26 /sbin/init
408 root 20 0 27708 2644 2340 S 0.0 0.5 0:00.17 /lib/systemd/systemd-journald
440 root 20 0 94772 1556 1380 S 0.0 0.3 0:00.00 /sbin/lvmtool -f
464 root 20 0 42488 3896 3104 S 0.0 0.8 0:00.05 /lib/systemd/systemd-udevd
1115 root 20 0 5220 144 36 S 0.0 0.0 0:00.04 /sbin/iscsid
1148 root 20 0 95368 1476 1344 S 0.0 0.3 0:00.00 /usr/bin/lxcfs /var/lib/lxcfs/
1153 root 20 0 95368 1476 1344 S 0.0 0.3 0:00.00 /usr/bin/lxcfs /var/lib/lxcfs/
1120 root 20 0 95368 1476 1344 S 0.0 0.3 0:00.00 /usr/bin/lxcfs /var/lib/lxcfs/
1122 root 20 0 4396 1264 1176 S 0.0 0.3 0:00.00 /usr/sbin/acpid
1124 daemon 20 0 26044 2208 2012 S 0.0 0.4 0:00.00 /usr/sbin/atd -f
1157 syslog 20 0 254M 3504 2872 S 0.0 0.7 0:00.01 /usr/sbin/rsyslogd -n
1158 syslog 20 0 254M 3504 2872 S 0.0 0.7 0:00.00 /usr/sbin/rsyslogd -n
1159 syslog 20 0 254M 3504 2872 S 0.0 0.7 0:00.01 /usr/sbin/rsyslogd -n
1127 syslog 20 0 254M 3504 2872 S 0.0 0.7 0:00.02 /usr/sbin/rsyslogd -n
1161 root 20 0 268M 6176 5480 S 0.0 1.2 0:00.03 /usr/lib/accountsservice/accounts-daemon
1175 root 20 0 268M 6176 5480 S 0.0 1.2 0:00.00 /usr/lib/accountsservice/accounts-daemon
1155 root 20 0 27728 3024 2760 S 0.0 0.6 0:00.00 /usr/sbin/cron -f
1176 root 20 0 28656 3160 2792 S 0.0 0.6 0:00.01 /lib/systemd/systemd-logind
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit
```



# ■ Usando la CLI - Comandos básicos

Procesos:

- Señales de control:

Señal	ID	Función
SIGHUP	1	Reinicia el proceso, recarga la configuración
SIGINT	2	Interrumpe el proceso, igual a Ctrl+C
SIGQUIT	3	Cierra el proceso
SIGKILL	9	Fuerza la finalización del proceso
SIGSEGV	11	Fuerza un Segmentation fault
SIGTERM	15	Solicita al proceso que finalice



# ■ Usando la CLI - Comandos básicos

## Control de Procesos:

- Cuando iniciamos un programa en la shell, éste queda en primer plano
- Para enviar señales a un programa en ejecución podemos usar CTRL+C (termina el proceso) y CTRL+Z (para el proceso).
- Para continuar en segundo plano se utiliza el comando *bg*
- Para continuar con la ejecución en primer plano se utiliza *fg*
- El comando *jobs* muestra las tareas de la sesión actual
- Para iniciar un comando directamente en segundo plano se añade el carácter “&” al final del comando
- Si queremos que un proceso no finalice al terminar la shell lo lanzamos con *nohup*





# ■ Índice

- Linux
  - Comandos básicos
  - **Usuarios y grupos**
  - Herramientas
  - Instalación de Software
  - Instalación automatizada y arranque
  - Kernel: módulos y tuning



# ■ Usando la CLI - Usuarios y grupos

- Actualmente los SO son multiusuario y multitarea
- El SO se encarga de: identificación (login), permisos de ejecución, mecanismos de seguridad de hardware, protección de ficheros.
- El mecanismo para implementarlo se basa en usuarios y grupos
- root es el Superusuario: tiene todos los permisos y privilegios para realizar cualquier acción en el sistema
- La información de usuarios y grupos se guarda en:
  - */etc/passwd*
  - */etc/group*
  - */etc/shadow*



# ■ Usando la CLI - Usuarios y grupos

*/etc/passwd:*

```
oracle:x:1021:1020:Oracle user:/data/network/oracle:/bin/bash
```

Diagram illustrating the fields of the `/etc/passwd` entry for the user `oracle`, with arrows pointing to the corresponding field numbers:

Field Number	Field Value
1	oracle
2	x
3	1021
4	1020
5	Oracle user
6	/data/network/oracle
7	/bin/bash

1. login o nombre de usuario
2. Password, una x indica que la contraseña se almacena en `/etc/shadow`
3. UID: cada usuario debe tener asignado un ID
4. GID: ID del grupo primario
5. Comentario opcional
6. Path del home del usuario
7. Shell por defecto



<https://www.cyberciti.biz/faq/understanding-etcpasswd-file-format/>

# ■ Usando la CLI - Usuarios y grupos

*/etc/passwd:*

```
vagrant@ubuntu:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
```



# ■ Usando la CLI - Usuarios y grupos

*/etc/group:*

```
cdrom:x:24:vivek,student13,raj
```

1	2	3	4

1. nombre del grupo
2. Password, generalmente no se usa.
3. GID: ID del grupo
4. lista de usuarios pertenecientes al grupo



# ■ Usando la CLI - Usuarios y grupos

*/etc/shadow:*

vivek:\$1\$fnfffc\$PgteyHdicpGOfffXX4ow#5:13064:0:99999:7:::

- |    |                   |   |   |   |   |   |   |   |
|----|-------------------|---|---|---|---|---|---|---|
|    | ↓                 |   | ↓ |   | ↓ | ↓ | ↓ | ↓ |
| 1. | Nombre de usuario | 1 | 2 | 3 | 4 | 5 | 6 |   |
- Nombre de usuario
  - Password encriptada: 3 campos: ID del método de encriptación, SALT (string aleatorio) y hash de la contraseña
  - Días desde el 01/01/1970 hasta que la contraseña se cambió
  - Número mínimo de días entre cambios de contraseña
  - Número máximo de días de validez de la contraseña
  - Número de días previo a expiración donde se avisará al usuario
  - Número de días después de expiración de contraseña donde la cuenta se deshabilitará
  - Días desde el 01/01/1970 hasta que la cuenta está deshabilitada

<https://www.cyberciti.biz/faq/understanding-etcshadow-file/>



# ■ Usando la CLI - Usuarios y grupos

## Gestión de usuarios:

- Creación de usuarios: *adduser* o *useradd*
- Modificación de un usuario: *usermod*, *chfn*, *chsh*, *chage*
- Eliminación de usuarios: *deluser* o *userdel*
- Para cambiar la contraseña de un usuario: *passwd*
- Añadir un grupo: *addgroup* o *groupadd*
- Modificar un grupo: *groupmod*
- Eliminar un grupo: *groupdel* o *delgroup*
- Para cambiar la contraseña de un grupo: *gpasswd*



# ■ Usando la CLI - Usuarios y grupos

## Obtener información:

- Para saber qué usuario somos podemos utilizar *whoami*
- Para obtener información de los grupos a los que pertenecemos usamos *groups*
- El comando *id* nos muestra tanto el usuario como los grupos
- Con *su* podemos cambiar de usuario
- Para saber qué usuarios hay conectados en la máquina en un determinado momento usamos *who* o *w*
- Con los comandos *write* o *wall* podemos enviar mensajes al resto de usuarios logueados





# ■ Usando la CLI - Usuarios y grupos - Permisos

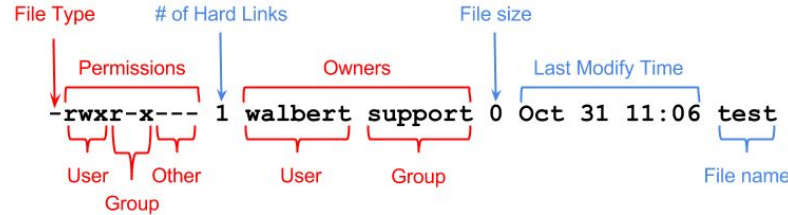
- Cada fichero del sistema debe pertenecer a un usuario y grupo
- Se debe especificar qué permisos tiene para cada uno de ellos
- Con el comando `ls -l` se puede ver la siguiente información:

```
-rw-r--r-- 1 user1 group1 0 Apr 28 11:18 file1.txt
```

- El primer campo indica los permisos del fichero
- El segundo campo indica el número de Hard Links del fichero o en el caso de directorios el número de directorios que contiene
- El tercer y cuarto campo indica el usuario y grupo al que pertenece el fichero



# Usando la CLI - Usuarios y grupos - Permisos



<https://www.ics.uci.edu/computing/linux/file-security.php>

- El file type nos indicará si es un fichero regular, un directorio o un dispositivo
- Los 9 caracteres de permisos se dividen en tres grupos: usuario, grupo y otros
  - `r` → permiso de lectura (read)
  - `w` → permiso de escritura (write)
  - `x` → permiso de ejecución



# ■ Usando la CLI - Usuarios y grupos - Permisos

## Representación octal vs simbólica

Octal	Binaria	Simbólica	Significado
0	000	---	Sin permiso
1	001	--x	Ejecución
2	010	-w-	Escritura
3	011	-wx	Escritura y ejecución
4	100	r--	Lectura
5	101	r-x	Lectura y ejecución
6	110	rw-	Lectura y escritura
7	111	rwX	Lectura, escritura y ejecución



# ■ Usando la CLI - Usuarios y grupos - Permisos

Modificación de permisos:

- *chmod* permite cambiar los permisos utilizando la notación octal o simbólica:  
*chmod go+w fichero1.txt*  
*chmod 566 fichero1.txt*
- *chown* permite cambiar el usuario de un determinado fichero
- *chgrp* permite cambiar el grupo de un fichero

Los permisos que se ponen por defecto al crear un fichero se controlan con el comando *umask*



# ■ Índice

- Linux
  - Comandos básicos
  - Usuarios y grupos
  - **Herramientas**
  - Instalación de Software
  - Instalación automatizada y arranque
  - Kernel: módulos y tuning



# ■ Usando la CLI - Herramientas

## **Variables de entorno** y de shell:

- Variables del Shell establecen información usada por uno o más programas.
- Las variables de entorno modifican en algunos casos estas variables
- Muchos programas usan estas variables para obtener información de sistema, almacenar datos o cargar configuraciones
- Tipos:
  - **Globales:** son visibles desde la shell y desde cualquier proceso hijo
  - **Locales:** únicamente están disponibles en la shell que las ha creado



# ■ Usando la CLI - Herramientas

## **Variables de entorno** y de shell:

- Para listarlas, establecerlas o modificarlas usamos:
  - *set*: imprime o modifica variables de shell
  - *printenv*: imprime las variables exportadas
  - *env*: lista las variables exportadas o ejecuta un programa en un entorno modificado
  - *export*: modifica una variable
  - *unset*: elimina una variable de entorno



# ■ Usando la CLI - Herramientas

## **Variables de entorno** y de shell:

- La definición implica definir primero la variable y luego exportarla al entorno global:  

```
$ VARIABLE="valor"
```

```
$ export VARIABLE
```
- Para variables locales por convención se usan palabras en mayúscula
- Cambiar el valor en una shell hija no implica que se cambie el valor en la shell padre





# ■ Usando la CLI - Herramientas

- Variables de entorno
  - Local

```
keepcoding@keepcoding-VirtualBox:~$ echo $MY_VAR
keepcoding@keepcoding-VirtualBox:~$ MY_VAR="local"
keepcoding@keepcoding-VirtualBox:~$ echo $MY_VAR
local
keepcoding@keepcoding-VirtualBox:~$ bash
keepcoding@keepcoding-VirtualBox:~$ echo $MY_VAR

keepcoding@keepcoding-VirtualBox:~$ exit
exit
```



# ■ Usando la CLI - Herramientas

- Variables de entorno
  - Global

```
keepcoding@keepcoding-VirtualBox:~$ MY_VAR="global"
keepcoding@keepcoding-VirtualBox:~$ echo $MY_VAR
global
keepcoding@keepcoding-VirtualBox:~$ export MY_VAR
keepcoding@keepcoding-VirtualBox:~$ bash
keepcoding@keepcoding-VirtualBox:~$ echo $MY_VAR
global
keepcoding@keepcoding-VirtualBox:~$ exit
exit
keepcoding@keepcoding-VirtualBox:~$ □
```



# ■ Usando la CLI - Herramientas

## **Variables de entorno** y de shell más útiles:

- *PWD*: directorio actual
- *BASH\_VERSION*: versión de bash que utilizamos
- *RANDOM*: genera un número aleatorio
- *SECONDS*: segundos desde que hemos abierto la shell
- *HOSTNAME*: nombre del sistema
- *OSTYPE*: tipo de sistema operativo que usamos
- *MACHTYPE*: arquitectura del sistema
- *HISTFILESIZE*: tamaño del fichero de histórico
- *HISTCMD*: número del comando actual en la historia
- *HISTFILE*: fichero donde se guarda la historia de comandos



# ■ Usando la CLI - Herramientas

**Variables de entorno** y de shell más útiles:

- *HOME*: directorio home del usuario
- *LANG*: determina el locale a usar
- *PATH*: lista de directorios donde la shell busca los comandos
- *PS1*: settings del prompt
- *SHELL*: path del shell que estamos corriendo actualmente
- *TERM*: tipo de terminal que tenemos actualmente
- *EDITOR*: editor por defecto

[Bash Variables \(Bash Reference Manual\)](#)



# ■ Usando la CLI - Herramientas

## Información básica del sistema:

- CPU: *lscpu*
- Memoria: *free*
- Discos: *df*, *lsblk*, *lsscsi*
- Dispositivos PCI: *lspci*
- Dispositivos USB: *lsusb*
- Ficheros abiertos: *lsdf*
- Todo el hardware: *lshw*
- Versión de la distribución: *lsb\_release*
- Cuánto tiempo lleva el sistema activo: *uptime*

[ls\\* Commands Are Even More Useful Than You May Have Thought](#)



# ■ Usando la CLI - Herramientas

## Programación de tareas:

- El comando *at* nos permite lanzar una acción más tarde especificando hora y fecha
- *atq* y *atrm* nos permite listar y borrar tareas programadas
- Para lanzar tareas periódicamente usamos el cron.
- Se pueden especificar tareas a ejecutar en los siguientes ficheros y directorios:
  - */etc/crontab*
  - */etc/cron.hourly/*
  - */etc/cron.daily/*
  - */etc/cron.weekly/*
  - */etc/cron.monthly/*
  - */etc/cron.d/*



# ■ Usando la CLI - Herramientas

Crontab y ficheros en /etc/cron.d:

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Usuarios sin privilegios pueden usar el comando `crontab -e` para programar ejecuciones. En este caso la definición queda definida en el fichero `/var/spool/cron/crontab/usuario`



# ■ Usando la CLI - Herramientas

## **Personalización** del sistema:

- Los shells tienen ficheros de arranque asociados
- Estos ficheros permiten definir:
  - código o funciones
  - variables
  - alias
- Cualquier usuario se puede configurar sus modificaciones:
  - `$HOME/.profile`
  - `$HOME/.bashrc`





# ■ Usando la CLI - Herramientas

Personalización del sistema:

- Se pueden añadir modificaciones a todos los usuarios:
  - `/etc/profile`
  - `/etc/profile.d/*`
  - `/etc/bash.bashrc`
- También se pueden modificar los perfiles para usuarios que se crearán posteriormente:
  - `/etc/skel/`



# ■ Usando la CLI - Herramientas

Personalización del sistema, ejemplos:

- PATH:

```
# set PATH so it includes user's private bin directories
PATH="$HOME/bin:$HOME/.local/bin:$PATH"
```

- PS1:

```
if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
```

- alias:

```
# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'
```



# ■ Usando la CLI - Herramientas

- prompt:
  - texto que aparece en el inicio de cada línea
  - formato: *usuario@hostname:directorio\$*
  - El último carácter suele indicar:
    - \$ → usuario normal
    - # → usuario superadministrador (root)
  - El formato se especifica en la variable PS1
  - PS1 Generator
    - <http://ezprompt.net/>
    - <http://bashrcgenerator.com/>



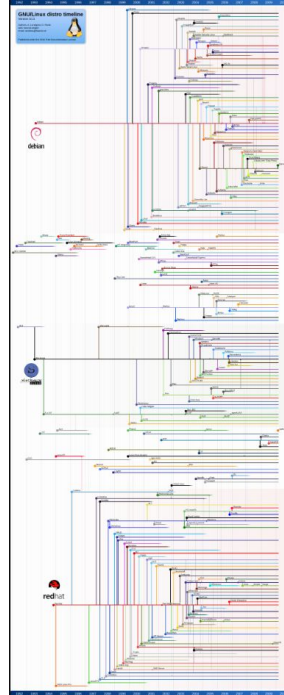
# ■ Índice

- Linux
  - Comandos básicos
  - Usuarios y grupos
  - Herramientas
  - **Instalación de Software**
  - Instalación automatizada y arranque
  - Kernel: módulos y tuning



# ■ Usando la CLI - Instalación de software

[OT] Historia de las distribuciones linux:



# ■ Usando la CLI - Instalación de software

DPKG:

Description	Command
Install package	<code>dpkg -i {package.deb}</code>
Install all packages from a directory structure	<code>dpkg -R {directory}</code>
Update package	<code>dpkg -i {package.deb}</code>
List installed packages	<code>dpkg -l</code>
List files in a package with their target paths	<code>dpkg -c {package.deb}</code>
List files provided by an installed package	<code>dpkg -L {package}</code>
Determine the package that owns a file	<code>dpkg -S {/path/to/file}</code>
Remove an installed package, but leave the config files	<code>dpkg -r {package}</code>
Remove an installed package including config files	<code>dpkg -P {package}</code>



# ■ Usando la CLI - Instalación de software

APT:

Description	Command
Install package	<code>apt-get install {package}</code>
Update package	<code>apt-get update {package}</code>
Remove an installed package, but leave the config files	<code>apt-get remove {package}</code>
Remove an installed package including config files	<code>apt-get remove --purge {package}</code>
Resynchronize the package index files	<code>apt-get update</code>
Upgrade the Linux system including security updates	<code>apt-get upgrade</code>
Upgrade the Linux distribution	<code>apt-get dist-upgrade</code>
Search packages by name	<code>apt-cache search {package}</code>



# ■ Usando la CLI - Instalación de software

RPM:

Description	Command
Install package	<code>rpm -i {package.rpm}</code>
Update package	<code>rpm -U {package.rpm}</code>
List installed packages	<code>rpm -qa</code>
List files provided by an installed package	<code>rpm -qpl {package.rpm}</code>
Determine the package that owns a file	<code>rpm -qf {/path/to/file}</code>
Remove an installed package, but leave the config files	<code>rpm -e {package.rpm}</code>





# ■ Usando la CLI - Instalación de software

YUM:

Description	Command
Install package	yum install {package}
Update package	yum update {package}
Remove an installed package, but leave the config files	yum remove {package}
Resynchronize the package index files	yum clean
Upgrade the Linux system including security updates	yum update
Upgrade only security updates	yum update --security
List installed packages	yum list installed
List available packages	yum list
Search packages by name	yum search {package}
Display list of group software	yum grouplist



# ■ Usando la CLI - Instalación de software

Configuración de repositorios:

- Debian y derivados:
  - `/etc/apt/sources.list`
  - `/etc/apt/sources.list.d`
- Red Hat y derivados
  - `/etc/yum.repos.d`



# ■ Usando la CLI - Instalación de software

Otras opciones de instalación:

- Paquete binario (java)

```
[root@centos ~]# mkdir /usr/java  
[root@centos ~]# cd /usr/java  
[root@centos java]# tar zxvf jre-8u73-linux-i586.tar.gz
```

- Github

<https://github.com/alerta/alerta>

- go

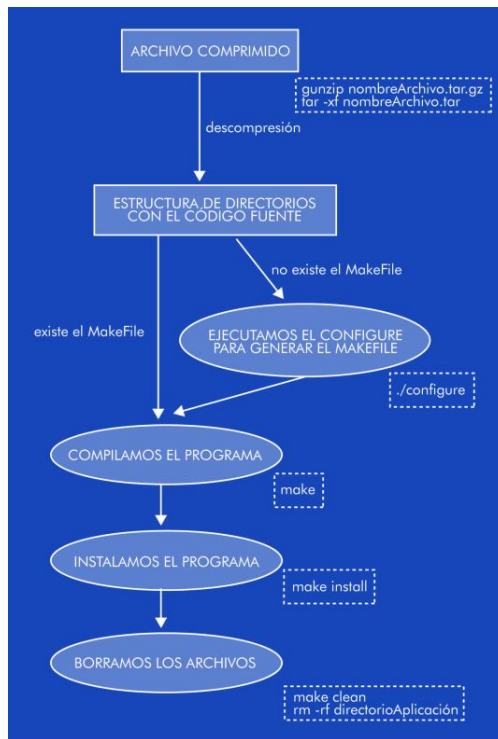
```
$ go get github.com/github/vulcanizer  
$ cd $HOME/go/src/github.com/github/vulcanizer/script  
$ ./build
```

- python: pip install ansible



# Usando la CLI - Instalación de software

## Compilar fuentes



# ■ Índice

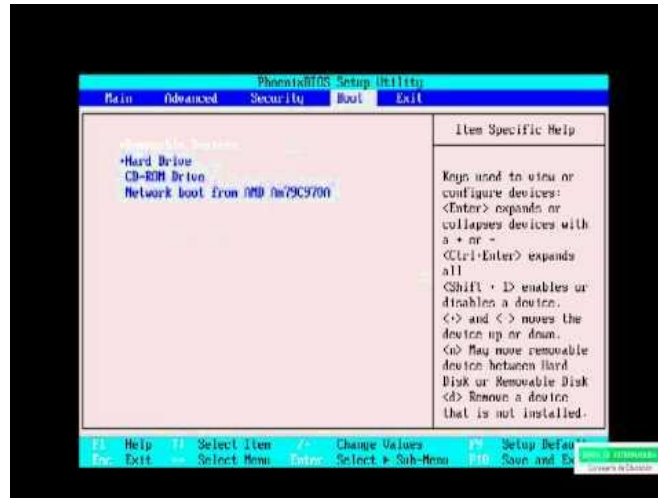
- Linux
  - Comandos básicos
  - Usuarios y grupos
  - Herramientas
  - Instalación de Software
  - **Instalación automatizada y arranque**
  - Kernel: módulos y tuning



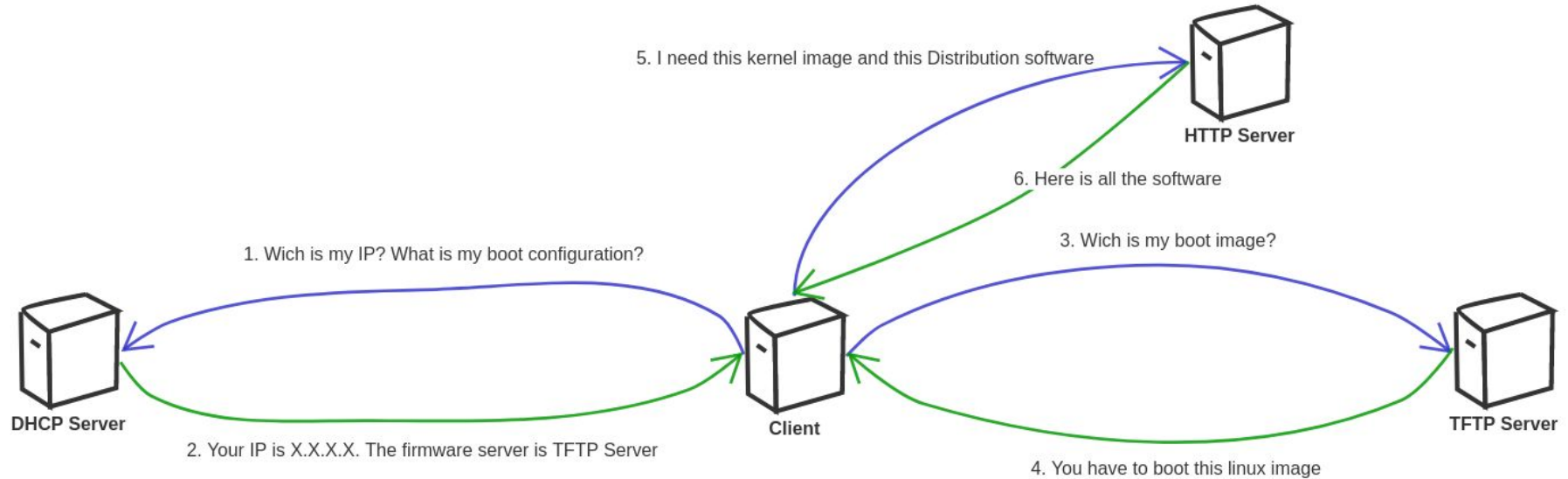
# ■ Usando la CLI - Instalación automatizada

Opciones de instalación:

- Desde una imagen bootable: DVD/USB/ISO
- Por red (PXE):



# Usando la CLI - Instalación automatizada



# ■ Usando la CLI - Instalación automatizada

Fases del proceso de instalación:

- Boot screen
- Selección de idioma y configuración del teclado
- Configuración de hora y geográfica
- Particionado del disco duro
- Configuración de red
- Selección e instalación de paquetes
- Configuración de usuarios
- Creación del disco de arranque e instalación del Bootloader





# ■ Usando la CLI - Instalación automatizada

Demo Packer

[www.packer.io/downloads](http://www.packer.io/downloads)

<https://github.com/geerlingguy/packer-boxes>



# ■ Usando la CLI - Arranque del Sistema Operativo

Proceso de arranque:

1. Bootloader: GRUB o LILO.
2. Se inicia el kernel y pasa a tener el control
3. Se inicia el hardware del sistema
4. Montaje de partición /
5. Activación del resto de particiones
6. Activación del init, configuraciones avanzadas de hardware y arranque de demonios y servicios

Con el comando *dmesg* se pueden ver los logs de arranque



# ■ Usando la CLI - Arranque del Sistema Operativo

Systemd:

- adoptado en las últimas versiones de las distribuciones convirtiéndose en el estándar de facto
- El proceso que inicia todo tiene como PID = 1
- Proporciona un sistema de arranque más rápido y más flexible
- La configuración reside en /etc/systemd y sus subdirectorios



# ■ Systemd

- Originalmente los sistemas Linux y UNIX arrancaban el proceso init
- Tras la adopción de SystemD por la mayoría de distribuciones el proceso de arranque viene gestionado por el proceso systemd
- systemd, por tanto, será el padre de todos los procesos y es el responsable de llevar a Linux al estado final deseado
- El estado deseado viene determinado en el fichero */etc/systemd/system/default.target*

```
[keepcoding@centos8 ~]$ ls -l /etc/systemd/system/default.target
lrwxrwxrwx. 1 root root 36 Apr  9 12:29 /etc/systemd/system/default.target -> /lib/systemd/system/graphical.target
```



# Systemd

- Targets

- Tienen una relación con los **runlevels** de SystemV:
  - Un runlevel especifica un estado del sistema operativo. Numerado del 0 al 6
  - Determina qué programas se pueden ejecutar tras el arranque

SystemV	Systemd Target	Alias	Descripción
	<i>halt.target</i>		<i>Para el sistema sin apagarlo físicamente</i>
0	<i>poweroff.target</i>	<i>runlevel0.target</i>	<i>Para el sistema y lo apaga físicamente</i>
S	<i>emergency.target</i>		<i>Single user. No se montan filesystems ni se arrancan servicios</i>
1	<i>rescue.target</i>	<i>runlevel1.target</i>	<i>Se montan los filesystems y se inicia una shell de rescate</i>
2		<i>runlevel2.target</i>	<i>Multiusuario sin NFS</i>
3	<i>multi-user.target</i>	<i>runlevel3.target</i>	<i>Multiusuario con todos los servicios arrancados en modo CLI</i>
4		<i>runlevel4.target</i>	<i>No se usa</i>
5	<i>graphical.target</i>	<i>runlevel5.target</i>	<i>Igual al multi-user con entorno gráfico</i>
6	<i>reboot.target</i>	<i>runlevel6.target</i>	<i>Reinicio</i>
	<i>default.target</i>		<i>Se debe crear un link simbólico a multi-user o graphical</i>



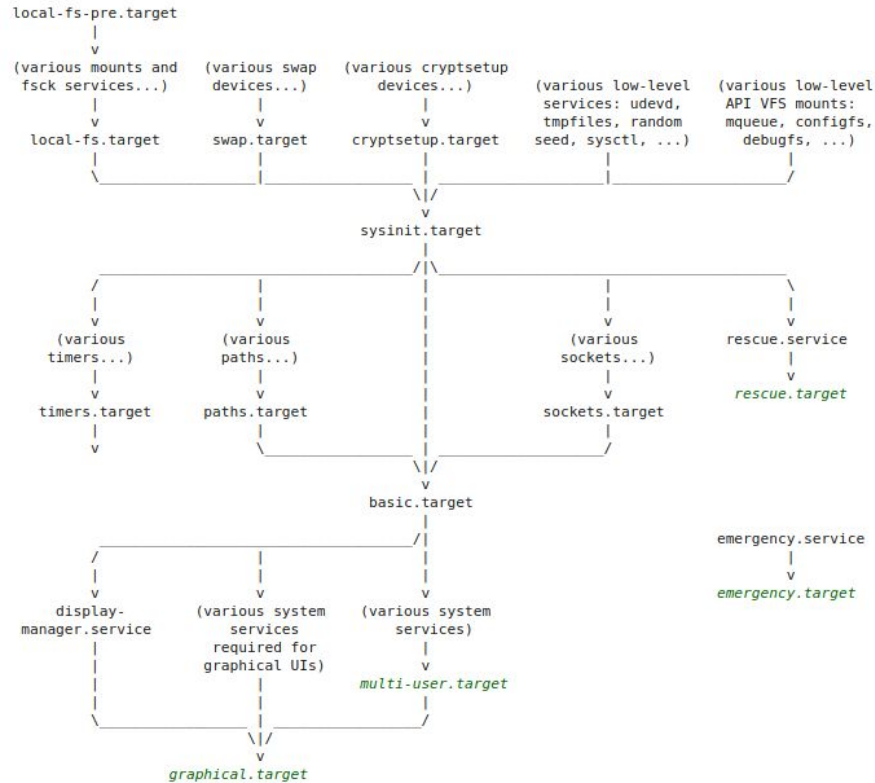
# Systemd

- Targets
  - Cada target tiene una serie de dependencias descritas en sus ficheros de configuración
  - Estas dependencias suelen ser los servicios requeridos para que Linux se ejecute con un específico nivel de funcionalidad
  - La relación completa de dependencias se puede consultar en el *“man bootup”*
  - Para cambiar de un target a otro usamos:  
*systemctl isolate {target}*
  - Para cambiar el target por defecto:  
*systemctl get-default*  
*systemctl set-default {target}*



# Systemd

- Targets



# ■ Gestionar servicios de SystemD

- Unidades
  - Definen los objetos o recursos que va a gestionar SystemD
  - Existen varios tipos:
    - service
    - path
    - socket
    - device
    - mount
    - automount
    - timer
    - swap





# ■ Gestionar servicios de SystemD

- Unidades
  - Para gestionar las unidades usaremos el comando `systemctl`
    - `systemctl list-units`
    - `systemctl list-unit-files`
    - `systemctl cat {unit}`
  - Las unidades vienen definidas en:
    - `/lib/systemd/system/`
    - `/etc/systemd/system/`



# ■ Gestionar servicios de SystemD

- Unidades
  - Cada unidad tiene su correspondiente fichero declarativo de definición
  - Este fichero está dividido en secciones:
    - Unit: información genérica sobre la unidad e independiente del tipo (descripción, documentación, dependencias)
    - Install: información de instalación usada cuando se habilita o deshabilita
    - Específica del tipo, p.e. Service, Socket



# ■ Gestionar servicios de SystemD

- Chequear estado
  - Para verificar el estado de una unidad usamos el comando `systemctl` con los siguientes comandos:
    - `status`
    - `is-active`
    - `is-enabled`
    - `is-failed`



# ■ Gestionar servicios de SystemD

- Dependencias
  - podemos listar las dependencias de una unidad con  
*systemctl list-dependencies {unit}*
- Enmascaramiento
  - Ayuda a evitar que una unidad se inicie por error
  - Evita conflictos entre servicios  
*systemctl mask {unit}*  
*systemctl umask {unit}*



# ■ Gestionar servicios de SystemD

- Iniciar/parar
  - para iniciar, parar, reiniciar un servicio o unidad lo haremos de la siguiente forma:

Comando	Descripción
<code>systemctl start {unit}</code>	Arranca el servicio
<code>systemctl stop {unit}</code>	Para el servicio
<code>systemctl restart {unit}</code>	Reinicia el servicio
<code>systemctl reload {unit}</code>	Recarga la configuración
<code>systemctl enable {unit}</code>	Habilita el servicio
<code>systemctl disable {unit}</code>	Deshabilita el servicio



# ■ Usando la CLI - Arranque del Sistema Operativo

## Systemd - Comandos avanzados:

Tips	Explanation
<code>systemd-cgls</code>	Recursively show control group contents in a tree
<code>systemctl set-default multi-user.target</code>	Change the default boot target
<code>systemctl show -p "Wants" multi-user.target</code>	show unit dependencies
<code>systemd --test --system --unit=foobar.target</code>	know what will be started in a specific target
<code>systemctl isolate graphical.target</code>	change the current target
<code>systemctl list-units --type=target</code>	show all targets that are currently active



# ■ Índice

- Linux
  - Comandos básicos
  - Usuarios y grupos
  - Herramientas
  - Instalación de Software
  - Instalación automatizada y arranque
  - **Kernel: módulos y tuning**



# ■ Usando la CLI - Kernel

- El Kernel es el corazón del Sistema Operativo
- Es la capa entre los procesos de usuario y el hardware disponible en la máquina
- Los procesos del Kernel (Kernel threads) son fácilmente reconocibles en la salida del comando *ps* al estar entre corchetes
- Se encarga de la inicialización del hardware mediante drivers
- Los drivers se cargan dinámicamente mediante módulos





# ■ Usando la CLI - Analizando el Kernel

- dmesg → muestra los mensajes de log del kernel desde el arranque del SO
- /proc filesystem → interfaz con el kernel. Contiene ficheros del estado actual del sistema
- uname → devuelve información respecto al SO

[OT]: para conocer qué versión del SO tiene nuestro sistema:

- ubuntu → /etc/lsb-release
- redhat → /etc/redhat-release



# ■ Usando la CLI - Módulos del Kernel

- En los inicios de Linux el kernel se debía compilar con todos los drivers necesarios para un sistema
- A partir de la versión 2 se evolucionó a un sistema modular
  - kernel relativamente pequeño
  - módulos que se cargan dinámicamente proporcionando drivers
  - solo se cargan los módulos necesarios para cada sistema
- Los módulos no solo cargan drivers si no cualquier otra funcionalidad que no esté en el kernel



# ■ Usando la CLI - Inicialización de Hardware

- En el arranque el kernel sondea el hardware disponible
- Una vez detectado el procesos systemd-udev se encarga de cargar el módulo correspondiente
- Para decidir cómo se inicializa un dispositivo se leen los ficheros de reglas `/usr/lib/udev/rules.d` o `/lib/udev/rules.d`
- Para hacer cambios en estas reglas se realizan en `/etc/udev/rules.d`
- Una vez se ha cargado el módulo se puede ver el estado en el `/sys` filesystem
- Con `udevadm monitor` podemos ver los eventos relacionados con el hardware



# ■ Usando la CLI - Gestión de módulos

- Si se tiene que gestionar un módulo de forma manual existe una serie de comandos:

Comando	Uso
lsmod	Lista los módulos de kernel cargados actualmente
modinfo	Muestra información sobre los módulos del kernel
modprobe	Carga un módulo de kernel incluyendo todas sus dependencias
modprobe -r	Desactiva un módulo de kernel considerando sus dependencias
lspci -k	Lista los dispositivos hardware con el módulo con el que se han cargado

- También se pueden cargar automáticamente módulos definiendolos en `/etc/modprobe.d`



# ■ Usando la CLI - Tuning

Cambiar el comportamiento del kernel

- `/etc/sysctl.conf`
- `sysctl -p`
- Cambio temporal → `sysctl -w variable=value`

```
$ man sysctl.conf
```

*The description of individual parameters can be found in the kernel documentation.*

Google → `fs.file-max site:kernel.org`





# ¿Preguntas?



GRACIAS

[www.keepcoding.io](http://www.keepcoding.io)

