



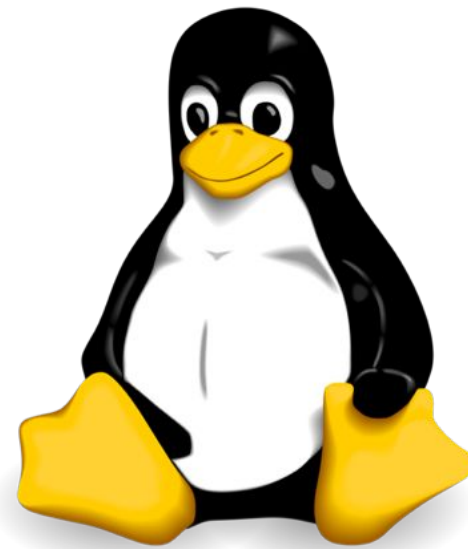
EXCUSE ME!





Do you have a moment to talk  
about Linux?





Tux



Linux



# ■ Historia

- En 1983, Richard Stallman inicia el proyecto GNU (GNU's Not Unix) para crear un sistema operativo Unix basado en software libre.
- **Linus Torvalds** creó el kernel de Linux en 1992 a partir de MINIX
- El kernel de Linux junto herramientas del proyecto GNU, generó la primera versión del sistema operativo GNU/Linux.
- A partir de esta combinación nacen varias de distribuciones:
  - Debian o basadas en Debian (Ubuntu, Linux Mint, etc.)
  - Red Hat o basadas en Red Hat (Fedora, SUSE, etc.)
  - Otras, pero las anteriores son las más importantes





ubuntu



# ■ Ubuntu

- Distribución creada a partir de Debian
- Su objetivo es hacer Debian más fácil de utilizar
- Soportada por Canonical, empresa de Mark Shuttleworth
- Dos releases anuales:
  - En Abril: YEAR.04 (14.04, 15.04, etc.)
  - En Octubre: YEAR.10 (14.10, 15.10, etc.)
- Tiene releases LTS: Long Term Support
- Cuenta con diferentes versiones: Desktop, Server, Education, etc.
- No tiene cuenta **root**





■ Empezemos por lo básico



# ■ El teclado

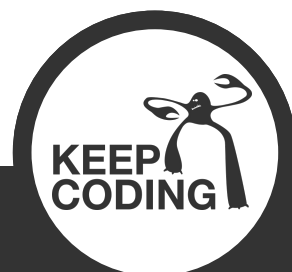




```
1. alberto@einstein: ~ (ssh)
top - 11:47:09 up 5 days, 11:23, 1 user, load average: 0.34, 0.20, 0.22
Tasks: 123 total, 1 running, 122 sleeping, 0 stopped, 0 zombie
%Cpu(s): 7.6 us, 0.7 sy, 0.0 ni, 84.8 id, 1.7 wa, 0.0 hi, 0.0 si, 5.3 st
KiB Mem: 1692588 total, 1675676 used, 16912 free, 17240 buffers
KiB Swap: 4194300 total, 1208192 used, 2986108 free. 117708 cached Mem

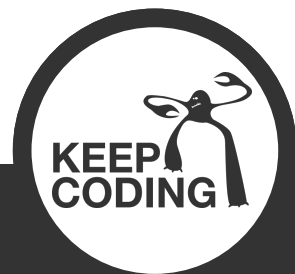
  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
 6881 www-data  20   0 382500 66704 35784 S   4.0   3.9   1:40.42 php5-fpm
16148 www-data  20   0 388432 72720 35876 S   3.7   4.3   1:55.54 php5-fpm
22533 www-data  20   0 398296 78748 35876 S   3.7   4.7   6:25.99 php5-fpm
19117 mysql     20   0 3465264 63412 2548 S   0.7   3.7  14:24.28 mysqld
 6954 root      20   0  96240   920   328 S   0.3   0.1   1:44.49 beam
29111 redis     20   0  43980   1596   608 S   0.3   0.1   4:27.54 redis-server
29500 git       20   0 1144772 299928 3560 S   0.3  17.7  10:12.34 ruby
    1 root      20   0   37336   1940   308 S   0.0   0.1   0:38.33 init
    2 root      20   0         0         0      0 S   0.0   0.0   0:00.12 kthreadd
    3 root      20   0         0         0      0 S   0.0   0.0   0:53.64 ksoftirqd/0
    4 root      20   0         0         0      0 S   0.0   0.0   0:00.00 kworker/0:0
    6 root      rt    0         0         0      0 S   0.0   0.0   0:00.08 migration/0
    7 root      rt    0         0         0      0 S   0.0   0.0   0:02.34 watchdog/0
    8 root       0  -20         0         0      0 S   0.0   0.0   0:00.00 cpuset
    9 root       0  -20         0         0      0 S   0.0   0.0   0:00.00 khelper
   10 root      20   0         0         0      0 S   0.0   0.0   0:00.00 kdevtmpfs
   11 root       0  -20         0         0      0 S   0.0   0.0   0:00.00 netns
   12 root      20   0         0         0      0 S   0.0   0.0   0:00.02 xenwatch
```

# El terminal: bash para los amigos



# ■ El terminal: bash para los amigos

- Los servidores linux no suelen disponer de interfaz gráfica
- La administración se realiza a través del intérprete de comandos
- El intérprete más utilizado en Linux es Bash
- Dispone de un pequeño lenguaje de programación para hacer scripts



■ Conectando



# ■ Conectando

- Generalmente, conectamos a servidores Linux de manera remota
- Para conexión remota, se utiliza el protocolo **SSH** (Secure SHell)
- Es muy seguro y permite también transferencia de archivos (**SFTP**: SSH File Transfer Protocol)
- Presente en Linux y Mac
- En Windows hay que usar PuTTY





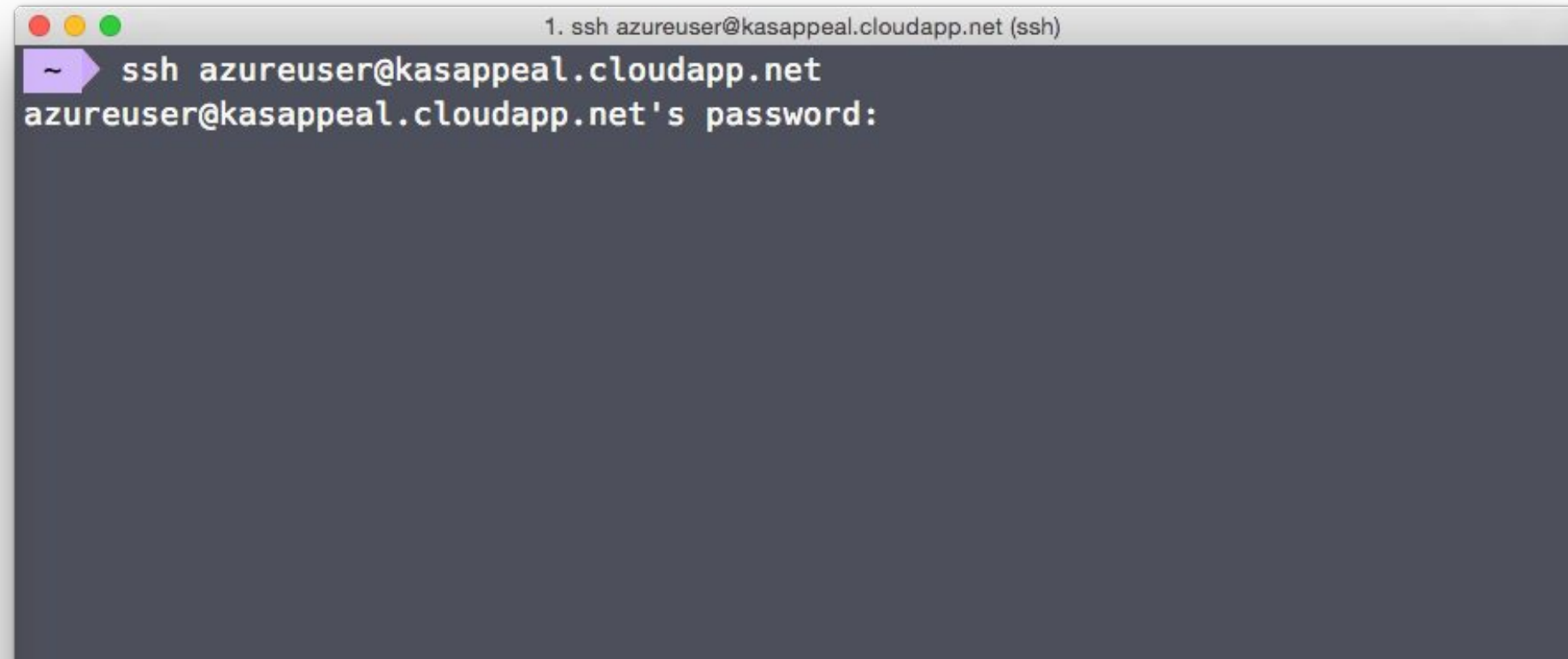
# ■ Conectando

El comando a utilizar:

```
ssh user@host
```



# ■ Conectando

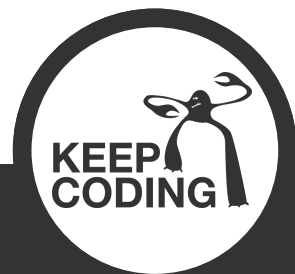


```
1. ssh azureuser@kasappeal.cloudapp.net (ssh)
~ ssh azureuser@kasappeal.cloudapp.net
azureuser@kasappeal.cloudapp.net's password:
```

Cuando Linux pregunta por una password y empezamos a escribir, en la pantalla no aparece nada. En realidad, se está escribiendo la password, pero no aparece en pantalla por seguridad (para que nadie la vea).



■ Desconectando



# ■ Desconectando

- Para desconectar podemos:
  - Usar el comando **logout**
  - Usar el comando **exit**
  - Pulsar **CTRL + D**



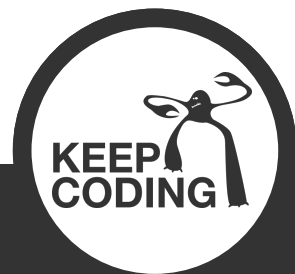


# ■ Haciéndonos amigos



■ ¿Quién está conectado?

who



■ ¿Qué día es hoy? ¿Qué hora es?

date



■ Muéstrame el calendario

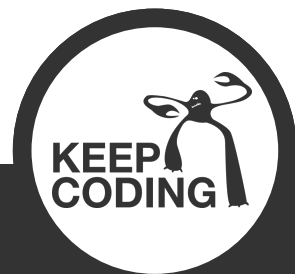
cal





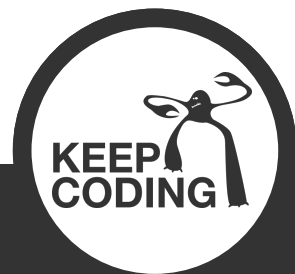
■ ¿En qué directorio estoy?

pwd

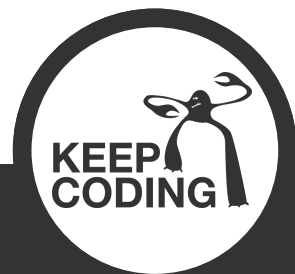


# ■ Cambiar mi contraseña

passwd



# ■ Dando un paseo por el sistema



# ■ Listar archivos de un directorio

## ls

Lista archivos del directorio local

## ls <path>

Lista archivos del directorio que indicamos

## ls -l <path>

Lista archivos del directorio que indicamos en modo lista





# ■ Cambiar de directorio

`cd ..`

Volver al directorio superior

`cd <path>`

Cambia del directorio actual al indicado



# ■ Rutas absolutas y relativas

■

Moverse al directorio actual (el . significa directorio actual)

**subdir      ./subdir**

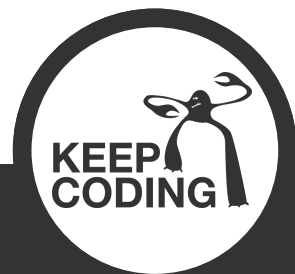
Ruta relativa al directorio actual (puede empezar por ./)

**/tmp/subdir**

Ruta absoluta a un directorio (empieza por /)



# ■ El sistema de archivos



# ■ El sistema de ficheros

- A diferencia de Windows, en Linux el sistema de ficheros no diferencia entre unidades (C:, D:, etc.)
- En las rutas se utiliza el slash “/” para indicar cambio de directorio (al contrario que en Windows, que es el backslash “\”)
- La raíz del sistema es “/” y a partir de ahí hay diferentes directorios
- No son necesarias las extensiones en los archivos



# ■ El sistema de ficheros

/  
/bin  
/boot  
/dev  
**/etc**  
**/home**  
/lib  
/lost+found  
/media  
/mnt

/opt  
/proc  
**/root**  
**/sbin**  
/srv  
/sys  
**/tmp**  
**/usr**  
**/var**



# ■ /etc

- Contiene ficheros de configuración del sistema
- Scripts que se ejecutan cuando arranca el sistema en:
  - `/etc/init.d/`
  - `/etc/rc.d/`



# ■ /home y /root

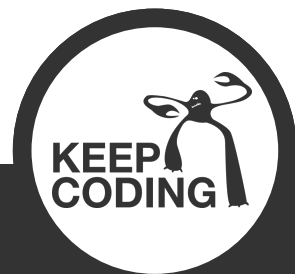
- Home de cada usuario del sistema:
  - /home/goku
  - /home/vegeta
  - /home/mutenroshi
- Cuando nos conectamos por SSH, por defecto accedemos a nuestro carpeta /home
- El /root es la casa de root (superadministrador)





■ /sbin

Almacena ejecutables de comandos que sólo pueden ser utilizados por un usuario super-administrador.





- Directorio temporal que permite a los usuarios almacenar datos
- Los datos serán eliminados cuando el sistema se reinicie o necesite espacio



# ■ /usr

Contiene subdirectorios con ejecutables del sistema.

- **/usr/bin:** ejecutables para cualquier usuario
- **/usr/lib:** librerías de programación (C/C++ habitualmente)
- **/usr/local:** archivos locales (desarrollados por ti)
- **/usr/sbin:** ejecutables sólo para administradores
- **/usr/share:** datos compartidos (documentación)
- **/usr/src:** código fuente del kernel de Linux



# ■ /var

Contiene **varios** subdirectorios y archivos con diferentes tipos de información

- **/var/log:** archivos de log
- **/var/mail:** buzones de e-mail de los usuarios
- **/var/run:** descriptores de procesos o sockets
- **/var/www:** archivos servidos por el servidor web



# ■ Jugando con archivos y directorios



# ■ Crear un directorio

```
mkdir <foldername>
```



# ■ Eliminar un directorio

```
rmmdir <foldername>
```

```
rm -rf <foldername>
```

Elimina recursivamente directorios y archivos.

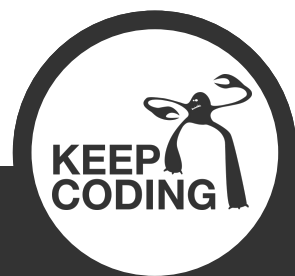
**PELIGRO:** rm -rf / como root borraría todo el disco





# ■ Crear un archivo vacío

```
touch <filename>
```



# ■ Editar un archivo

`nano <filename>`

`vi <filename>`

A veces `nano` no está disponible en algunas distribuciones por defecto. Sin embargo, `vi` sí suele estar instalado siempre, pero es más difícil de usar



## ■ Ver el contenido de un archivo

`cat <filename>`

Muestra el contenido de un fichero (ideal para archivos pequeños)

`more <filename>`

Permite leer un fichero grande poco a poco



# ■ Ver el contenido de un archivo

## head <filename>

Muestra las primeras líneas del fichero

## tail <filename>

Muestra las últimas líneas del fichero

## tail -f <filename>

Muestra en tiempo real el último contenido del fichero



# ■ ¿Cuántas palabras/líneas tiene?

**wc <filename>**

Cuántas palabras tiene un archivo de texto

**wc -l <filename>**

Cuántas líneas tiene un archivo de texto



# ■ ¿Qué diferencias hay?

```
diff <file1> <file2>
```

Muestra las diferencias entre dos archivos



## sort <filename>

Ordena las líneas de un archivo de texto





# ■ Filtra contenidos

grep <filename> <query>

Busca <query> en <filename>. Permite expresiones regulares.



# ■ Copiando archivos o directorios

```
cp <source> <target>
```

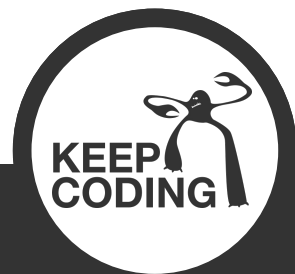
Copia un archivo de un sitio a otro



# ■ Mover/renombrar

```
mv <source> <target>
```

Mueve o renombra un archivo



# ■ Eliminar un archivo

```
rm <filename>
```

```
rm -f <filename>
```

Para que no pregunte si estamos seguros de eliminar



# ■ Alias/accesos directos

`ln <source> <target>`

Crea un alias del archivo. Si eliminamos el alias se elimina el archivo.

`ln -s <source> <target>`

Crea un acceso directo. Si eliminamos el acceso directo, **NO** elimina el archivo.  
También conocido como enlace simbólico.



# ■ Buscar archivos

```
find . -name <filename>
```

Busca un archivo de nombre filename en el directorio actual y subdirectorios.

```
find / -name <filename>
```

Busca un archivo de nombre filename en todo el sistema

<http://www.tecmint.com/35-practical-examples-of-linux-find-command/>



■ Ayuda



# BITCH



# READ THE DOCUMENTATION

memegenerator.net



`man <command name>`

Muestra el manual del programa o comando



# ■ Usuarios, grupos y permisos



# ■ Usuarios y grupos

- Linux es un sistema multiusuario
- Los usuarios no sólo son personas, también pueden ser servicios (como el servidor web, servidor de correo, etc.)
- Las cuentas de usuario, pueden pertenecer a uno o varios grupos
- Hay unos grupos especiales de administrador (**admin** o **sudo**)
- Suele haber una cuenta de super-administrador: **root**
  - En Ubuntu, el login con **root** está deshabilitado por seguridad



# ■ Permisos de archivos

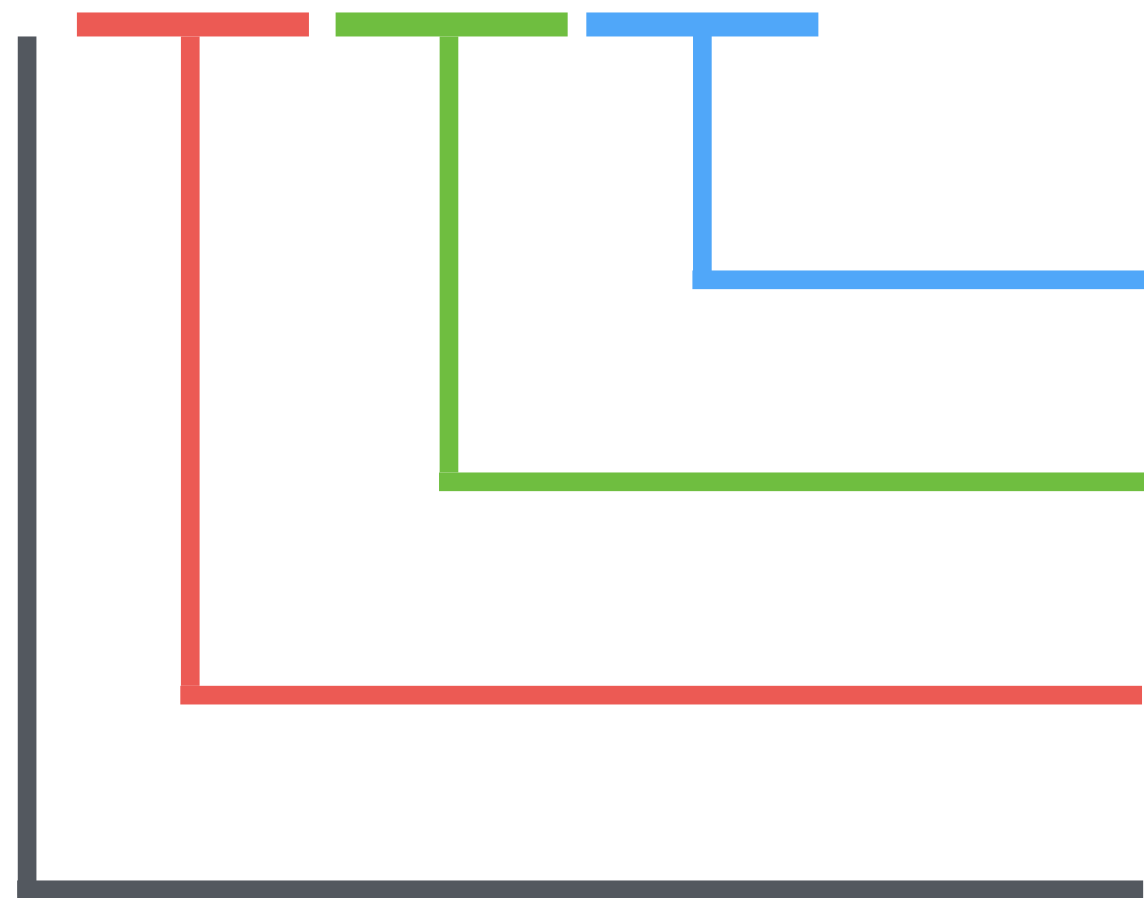
- Cada archivo o carpeta pertenece a un usuario y un grupo
- Los permisos de Linux permiten definir tres tipos de acceso:
  - Lectura
  - Escritura y/o borrado
  - Ejecución (para archivos o scripts ejecutables)
- Estos tres tipos de acceso, se definen en tres niveles:
  - Propietario del archivo o carpeta
  - Usuarios del mismo grupo del archivo o carpeta
  - Otros usuarios fuera del grupo



# ■ Ejemplo

Al ejecutar `ls -l`, en el listado nos muestra los permisos de un archivo

- `rwXrwxrwx` user group index.html



Permisos para otros usuarios

Permisos para usuarios del mismo grupo

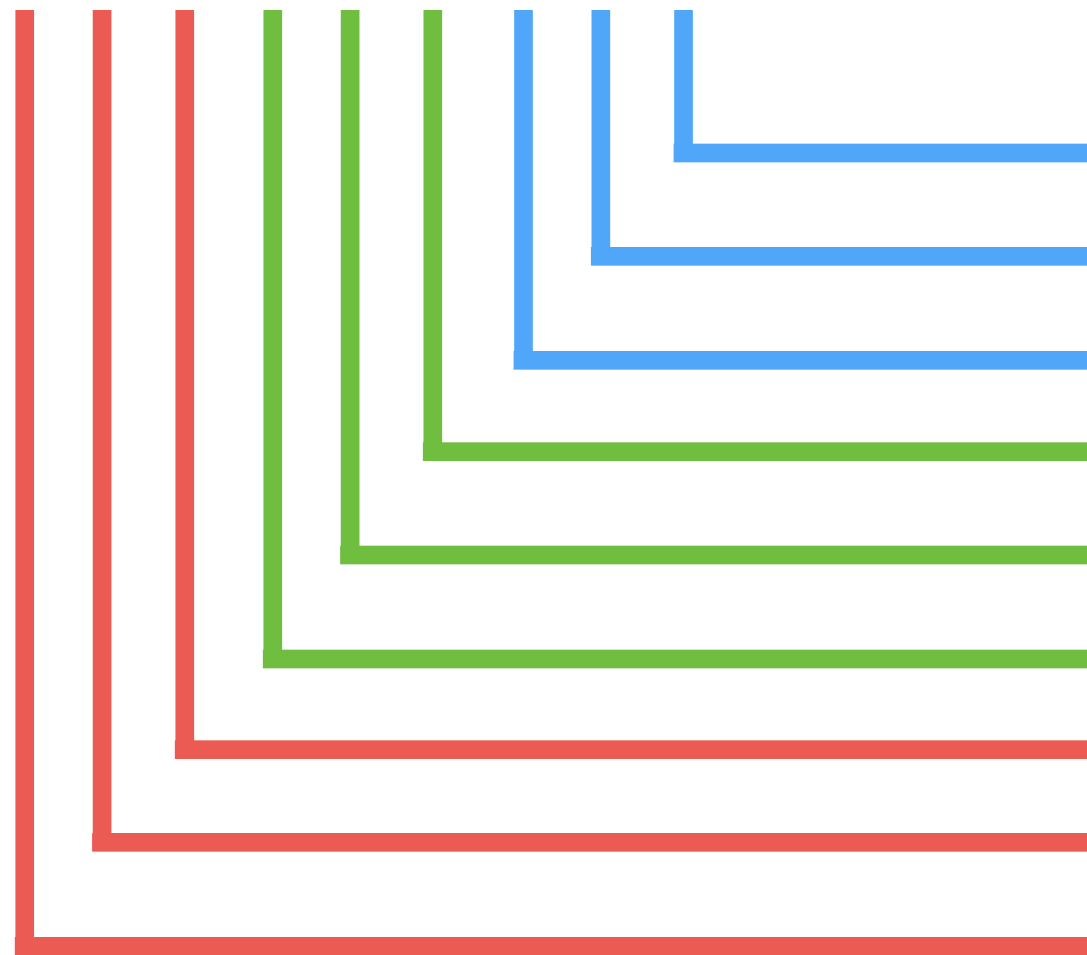
Permisos para el propietario

Se usa para indicar con si es un directorio (con `d`) o acceso directo (con `l`)



# ■ Ejemplo: archivo promiscuo

- **rwxrwxrwx** user group index.html

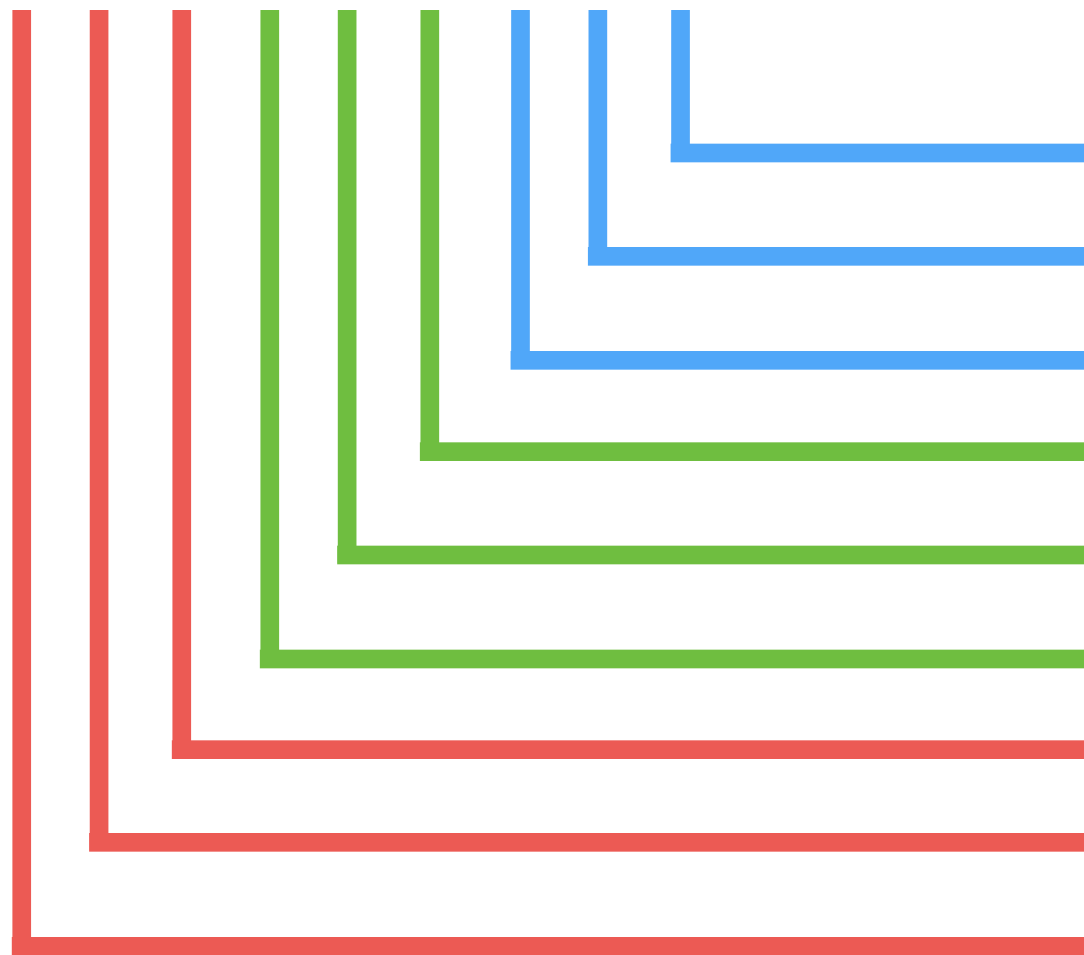


- Cualquier otro usuario puede ejecutar
- Cualquier otro usuario puede escribir/borrar
- Cualquier otro usuario puede leer
- Cualquier usuario del grupo puede ejecutar
- Cualquier usuario del grupo puede escribir/borrar
- Cualquier usuario del grupo puede leer
- El propietario puede ejecutar
- El propietario puede escribir/borrar
- El propietario puede leer



# ■ Ejemplo: no ejecutable

- - - - - user group index.html



Cualquier otro usuario **no** puede ejecutar

Cualquier otro usuario **no** puede escribir/borrar

Cualquier otro usuario **no** puede leer

Cualquier usuario del grupo **no** puede ejecutar

Cualquier usuario del grupo **no** puede escribir/borrar

Cualquier usuario del grupo **no** puede leer

El propietario **no** puede ejecutar

El propietario **no** puede escribir/borrar

El propietario **no** puede leer



# ■ Modificar permisos de archivos y carpetas

`chmod <target>±rwx <filename>`

`<target> = a / u / g / o`

`a = all / u = user/owner / g = group / o = other`





## ■ Ejemplo

**chmod u+wx backup**

Da (+) todos (rwx) los permisos para el propietario (u)

**chmod a+x backup**

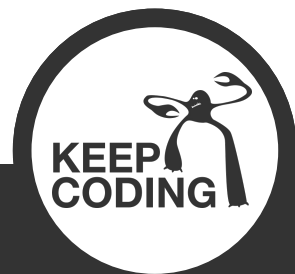
Da (+) permisos de ejecución (x) para todos (a)

**chmod g+rx backup**

Da (+) permisos de lectura y ejecución (rx) a los usuarios del grupo (g)

**chmod o-rw backup**

Quita (-) permisos de lectura y escritura (rw) a los otros usuarios (o)



# ■ Crear usuario

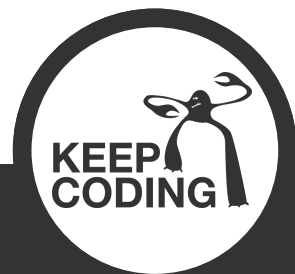


`adduser <username>`

Crear un usuario y un grupo con el mismo nombre del usuario y mete al usuario en dicho grupo.  
Esto lo registra en `/etc/passwd` y `/etc/group`



# ■ Ejecutando como administrador



# ■ Ejecutando como administrador



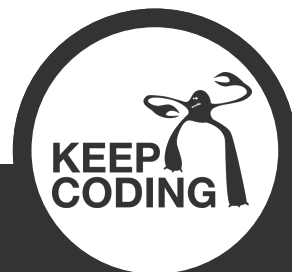
En necesario que seas administrador o sudoer



**SI EN UBUNTU NO PUEDO HACER LOGIN  
COMO ROOT**



**¿CÓMO EJECUTO COMANDOS COMO ADMINISTRADOR?**



# ■ Ejecutar como administrador

`sudo <command>`

Ejecuta lo que viene seguido de sudo como administrador.

El usuario debe tener permiso de **sudoer**.







# ■ Eliminar usuario



`deluser <username>`

Elimina un usuario del sistema.  
Lo elimina de `/etc/passwd`





# ■ Crear grupo



`addgroup <groupname>`

Crea un grupo añadiéndolo a  
`/etc/group`



# ■ Eliminar grupo



`delgroup <groupname>`

Elimina un grupo de `/etc/group`  
No elimina los usuarios de ese grupo



# ■ Añadir un usuario a un grupo

`adduser <username> <groupname>`



Añade el usuario al grupo en  
`/etc/group`



# ■ Permitir a un usuario hacer sudo



`adduser <username> sudo`

Añade el usuario al grupo sudo en  
`/etc/group`

Si eres sudo, tienes permisos para todo, da igual los permisos del archivo



# ■ Cambiar el propietario de un archivo o directorio

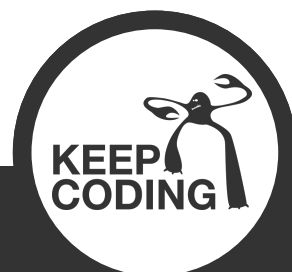
**chown <newowner> <filename>**

El archivo <filename> pasa a pertenecer a <newowner>

**chown -R <newowner> <foldername>**

Cambiar el propietario de <foldername> y todos sus archivos y subdirectorios.

Es un cambio recursivo (-R)



# ■ Cambiar el grupo de un archivo o directorio

## chgrp <newgroup> <filename>

El archivo <filename> pasa a pertenecer a <newgroup>

## chgrp -R <newgroup> <foldername>

Cambiar el grupo de <foldername> y todos sus archivos y subdirectorios.

Es un cambio recursivo (-R)



# ■ Cambiar el propietario y grupo a la vez

`chown <user>:<group> <filename>`



# ■ Cambiar mi contraseña

passwd





# ■ Cambiar contraseña de un usuario

passwd <username>



■ Apagar y reiniciar



# ■ Apagar y reiniciar

## shutdown

Apaga el servidor

## reboot

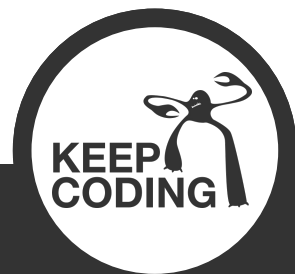
Reiniciar el servidor



En necesario que seas administrador o sudoer



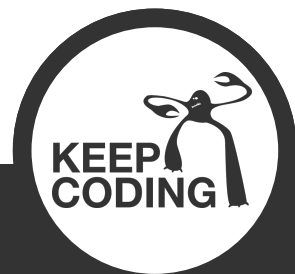
# ■ Instalando y desinstalando software



# Como en Linux todo es open source



# para instalar software



# hay



# que





# descargar



el



# código



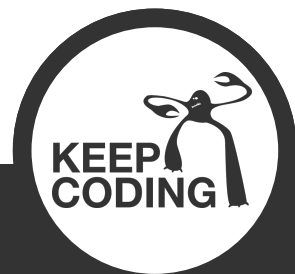
# fuentes



y



# compiler





tranquilos, eso era antes









# ■ Gestor de paquetes

En las distribuciones basadas en Debian (como Ubuntu), se puede instalar y desinstalar software a través del gestor de paquetes:

## apt-get

Este gestor se descarga paquetes ya compilados desde Internet para tu distribución y los instala en el sistema (descargando también otras dependencias si hiciera falta).



# ■ Instalar



`apt-get install <package>`

Instala el paquete <package> en el sistema



# ■ Desinstalar



`apt-get purge <package>`

Desinstala el paquete <package> del sistema



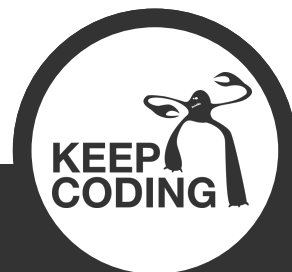


# ■ Actualizar repositorios



apt-get update

Actualiza los repositorios de software para buscar actualizaciones



# ■ Actualizar software

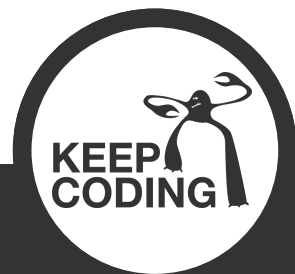


apt-get upgrade

Instala todas las actualizaciones de paquetes



# ■ Gestión de procesos



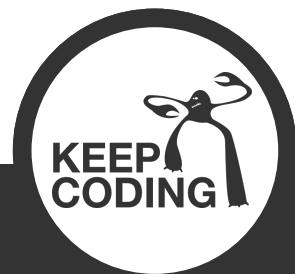


# ■ top: el CTRL+ALT+SUPR de Linux

top

Muestra en tiempo real información de los procesos ordenados por consumo de CPU (de mayor a menor).

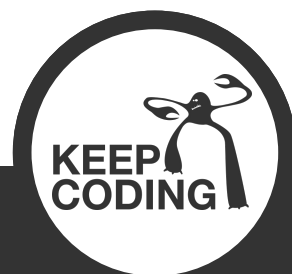
Para salir, pulsar la letra Q (de quit)



# ■ top: el CTRL+ALT+SUPR de Linux

```
1. alberto@einstein: ~ (ssh)
top - 11:47:09 up 5 days, 11:23,  1 user,  load average: 0.34, 0.20, 0.22
Tasks: 123 total,  1 running, 122 sleeping,  0 stopped,  0 zombie
%Cpu(s):  7.6 us,  0.7 sy,  0.0 ni, 84.8 id,  1.7 wa,  0.0 hi,  0.0 si,  5.3 st
KiB Mem:  1692588 total, 1675676 used,   16912 free,   17240 buffers
KiB Swap: 4194300 total, 1208192 used, 2986108 free. 117708 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
6881	www-data	20	0	382500	66704	35784	S	4.0	3.9	1:40.42	php5-fpm
16148	www-data	20	0	388432	72720	35876	S	3.7	4.3	1:55.54	php5-fpm
22533	www-data	20	0	398296	78748	35876	S	3.7	4.7	6:25.99	php5-fpm
19117	mysql	20	0	3465264	63412	2548	S	0.7	3.7	14:24.28	mysqld
6954	root	20	0	96240	920	328	S	0.3	0.1	1:44.49	beam
29111	redis	20	0	43980	1596	608	S	0.3	0.1	4:27.54	redis-server
29500	git	20	0	1144772	299928	3560	S	0.3	17.7	10:12.34	ruby
1	root	20	0	37336	1940	308	S	0.0	0.1	0:38.33	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.12	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:53.64	ksoftirqd/0
4	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0
6	root	rt	0	0	0	0	S	0.0	0.0	0:00.08	migration/0
7	root	rt	0	0	0	0	S	0.0	0.0	0:02.34	watchdog/0
8	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	cpuset
9	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	khelper
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
11	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	netns



# ■ Ver todos los procesos ejecutándose

## ps aux

Muestra todos los procesos en ejecución con mucha información (no sólo los que más CPU consumen, como hace top)

## ps aux | more

Para poder ir viéndolos poco a poco

## ps aux | grep <command>

Para buscar un proceso <command>



# ■ Matar un proceso

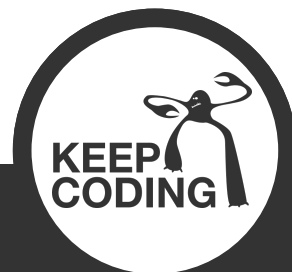
## kill <pid>

Mata el proceso número <pid>.  
Sólo vale para procesos nuestros.



## sudo kill <pid>

Para poder matar procesos de otros usuarios



# ■ Gestionando el espacio en disco



# ■ Espacio libre

`df`

Muestra el espacio libre de las particiones en bytes

`df -h`

Muestra el espacio libre legible para humanos (h)





# ■ Espacio ocupado

`du <folder>`

Muestra lo que ocupa cada archivo dentro de <folder>

`du -ch <folder>`

Más rápido (c) y entendible para humanos (h)



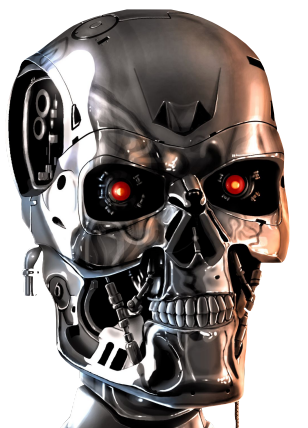
# ■ Seguridad





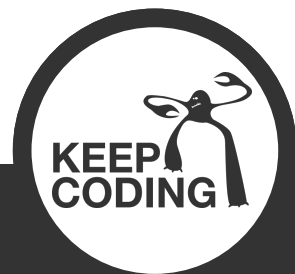
# ■ Consejos de seguridad

- Cambiar las contraseñas como mucho trimestralmente
- Cambia los puertos por defecto de los servicios que puedas
- Oculta los números de versión de los servicios
- Protege con SSL y autenticación HTTP los accesos de administración de tus plataformas
- No uses /admin /adm /cms /backend /backoffice (sé original)
- Usa un firewall
- Haz copias de seguridad diarias...y haz ensayos de recuperación
- Mantén actualizado tu software
- Elimina los humanos: son la mayor brecha de seguridad



# ■ fail2ban

- Es un servicio que se encarga de proteger el sistema sobre ataques a diferentes servicios mediante monitorización de los logs (entre otros aspectos).
- Cuando detecta una amenaza, reacciona baneando la IP del posible atacante.
- A veces se pasa de protector y nos deja sin acceso a nosotros
- Es muy sencillo de configurar



# ■ Instalación fail2ban

```
sudo apt-get install fail2ban
```



# ■ Fichero de configuración

/etc/fail2ban/jail.conf

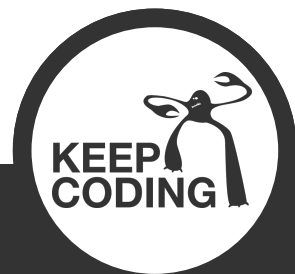


# ■ Configuración de filtros y acciones

/etc/fail2ban/filter.d/



# ■ Comprimir y descomprimir



# ■ Comprimir y descomprimir

Hay varias opciones, pero lo más habitual es usar **tar + gzip**:

- **tar** es un comando que empaqueta (junta varias carpetas en un archivo) pero no comprime
- **gzip** es un comando que comprime archivos (similar a zip)
- Se utiliza tar con la opción **-z** para comprimir en **gzip**
- También podemos usar **zip**, pero en Linux se usa mas **.tar.gz**

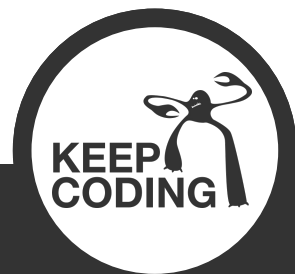


# ■ Comprimir en .tar.gz

```
tar -czvf <foo>.tar.gz <folder>
```



Comprime el directorio <folder> en un archivo  
<foo>.tar.gz



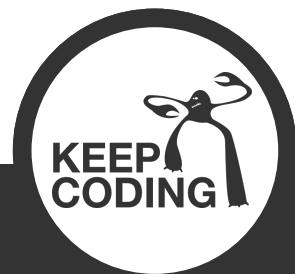


# ■ Descomprimir .tar.gz

```
tar -xzf <foo>.tar.gz
```



Descomprime <foo>.tar.gz en el directorio local



# ■ Comprimir en .zip

```
zip -r <foo>.zip <folder>
```

Comprime el directorio <folder> en un archivo <foo>.zip

ACHTUNG: `zip` no suele venir instalado en Linux.



# ■ Descomprimir en .zip

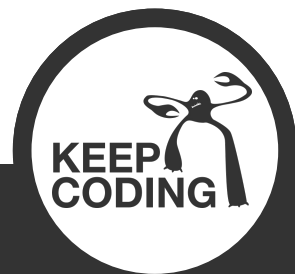
```
unzip <foo>.zip
```

Descomprime el archivo <foo>.zip en el directorio actual.

ACHTUNG: unzip no suele venir instalado en Linux.



# ■ Redirección



# ■ Redirección

En Bash podemos redirigir la salida de los comandos para que, en lugar de mostrarnos el resultado por pantalla, nos lo almacenen en un fichero.

También podemos redirigir la entrada, para que un comando en lugar de esperar un dato por pantalla (o teclado) directamente lo tome de un fichero.



## ■ Redirección de salida

```
echo "hi" > hello.txt
```

Guarda la salida del comando “echo” en el fichero “hello.txt”.

Si el fichero existe, machaca su contenido.

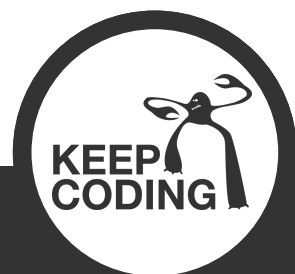
Si no existe el fichero, lo crea.

```
echo "hi" >> hello.txt
```

Igual que el anterior, pero:

Si el fichero existe, añade el contenido al final.

Si no existe el fichero, lo crea.



# ■ Redirección de entrada

```
patch index.js < v1.patch
```

Pasa el contenido de “hello.txt” como argumento al comando hello.



# ■ Pipes

Los pipes nos permiten pasar la información de un comando a otro como si fueran filtros. La salida de un comando se transforma en la entrada de otro comando.

```
cat quixote.txt | grep Rocinante | wc -l
```

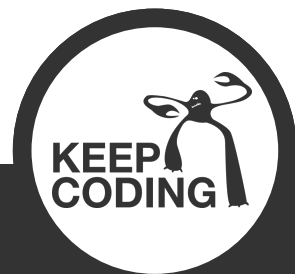
`cat` saca todo el contenido del fichero “quixote.txt” y se lo pasa como entrada al comando “**grep Rocinante**” que filtrará las líneas en las que aparece la palabra “Rocinante” y éste le pasa dicho resultado al comando “**more**” que nos permite ir leyendo poco a poco los resultados.





<http://www.commandlinefu.com/>

Repositorio de comandos



# ■ Scripting



# ■ Scripting

- La bash permite almacenar comandos en un fichero de texto y ejecutarlos como si fuera programas.
- Proporciona incluso instrucciones de control de flujo y bucles
- Los ficheros deben tener permisos de ejecución para poder ser tratados como programas.
- En la primera línea, se ha de indicar cuál es el intérprete que se debe utilizar.
- Podemos utilizar otros lenguajes de scripting (python, php, perl...)



# ■ Ejemplo

```
#!/bin/bash  
# This is a comment  
echo "My command is called $0\n"  
echo "The first argument is $1\n"  
for item in "$(ls)"  
do  
    echo "- $item\n"  
done
```



# ■ Primera línea

`#!/bin/bash` para scripts bash o shellscript

`#!/bin/python` para scripts python

`#!/bin/php` para scripts php



# ■ Variables implícitas

`$@` lista de parámetros recibida por el comando

`$0` nombre del propio comando/archivo

`$1` primer parámetro

`$2` segundo parámetro

...

`$n` n-ésimo parámetro

`$?` resultado de la salida del último comando (0 es OK, !=0 es KO)

`$$` número de proceso que se ejecuta

`$#` número de parámetros recibidos por el comando



# ■ Definición de variables

name="hello"

counter=0



# ■ Uso de variables

```
echo "Hello $name"
```

```
echo "Hello ${name}"
```

```
echo "$counter times"
```

```
echo "${counter} times"
```





# ■ Usando la salida de un comando

```
files=$(ls) # Ejecuta el comando ls y guarda su salida en "files"
```

```
echo $files
```



# ■ Condicionales

if <condition>

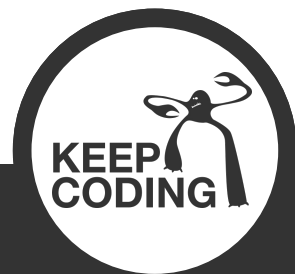
then

<stuff>

else

<otherstuff>

fi



# ■ Condiciones de las condicionales

if cp \$source \$target # si se ejecuta bien cp

if test -f \$source # si \$source es un fichero

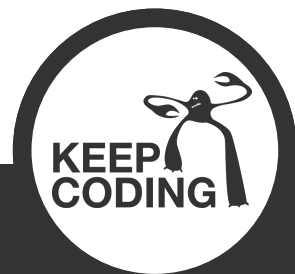
if [ -f "\$source" ] # si \$source es un fichero

if [ -d "\$source" ] # si \$source es un directorio

if [ -e "\$source" ] # si \$source existe

if \$source = \$target # si \$source es igual a \$target

if \$source != \$target # si \$source es distinto a \$target



# ■ Condiciones de las condicionales: numéricas

if \$numA -eq \$numB # \$numA == \$numB

if \$numA -ne \$numB # \$numA != \$numB

if \$numA -ge \$numB # \$numA >= \$numB

if \$numA -gt \$numB # \$numA > \$numB

if \$numA -le \$numB # \$numA <= \$numB

if \$numA -lt \$numB # \$numA < \$numB



# ■ Case

```
case <variable> in  
  <value1>  
    <stuff>  
;;  
<value2> | <value3>  
  <otherstuff>  
;;  
)  
  <anyotherstuff>  
;;  
esac
```



# ■ Bucle for

```
for file in $(ls)
```

```
do
```

```
<stuff>
```

```
done
```



# ■ Salida

Cuando nuestro programa acaba bien, debe devolver

`exit 0`

Si algo va mal, deberemos devolver un número distinto de cero (lo suyo es diferenciar los tipos de error con números).



# ■ Funciones

```
function <name> ()  
{  
    <commands>  
    return <int>  
}
```

```
<name> ()  
{  
    <commands>  
    return <int>  
}
```





# ■ Ejecutar en background



# ■ Ejecutar en background sin desconexión

<command> &

**ACHTUNG:** No vale para ejecutar en segundo plano y desconectarnos.  
El sistema puede matar el proceso pasado un tiempo.



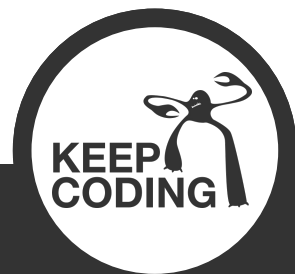
# ■ Ejecutar en background con desconexión

`nohup <command> &`

La posible salida que pueda sacar el comando, la almacenará en un archivo llamado `nohup.out`



# ■ Programando tareas automáticas



# ■ El cron

- Todos los sistemas Linux incluyen un programador de tareas: **cron**
- Nos permite programar comandos que se ejecuten automáticamente
- Básicamente, se almacena en un fichero la información de programación y el comando a ejecutar
- Como máximo podemos ejecutar repetitivamente hasta una vez por minuto



# ■ Programando tareas para mi usuario

crontab -e



# ■ El fichero de texto



Se puede utilizar el \* como comodín. Significa en cualquier valor.



# ■ Programando tareas de sistema



`sudo nano /etc/crontab`





# ■ El fichero de texto



Se puede utilizar el \* como comodín. Significa en cualquier valor.



# ■ Ejemplo crontab -e

```
30 10 * * 1 /usr/bin/who >> /home/who.txt
```

Todos los lunes a las 10:30

```
0,30 * * * 1 /usr/bin/who >> /home/who.txt
```

Todos los lunes a en punto o a y media de todas las horas

```
*/15 * * * * /usr/bin/who >> /home/who.txt
```

Cada 15 minutos

```
30 21 * * 6 /sbin/shutdown -h now
```

Apaga el servidor los sábados a las 21:30. **Sólo lo podría hacer root.**

[https://es.wikipedia.org/wiki/Cron\\_\(Unix\)](https://es.wikipedia.org/wiki/Cron_(Unix))



# ■ Ejemplo /etc/crontab

```
30 10 * * 1 larry /usr/bin/who >> /home/who.txt
```

Todos los lunes a las 10:30

```
0,30 * * * 1 steve /usr/bin/who >> /home/who.txt
```

Todos los lunes a en punto o a y media de todas las horas

```
*/15 * * * * sergei /usr/bin/who >> /home/who.txt
```

Cada 15 minutos

```
30 21 * * 6 root /sbin/shutdown -h now
```

Apaga el servidor los sábados a las 21:30. **Sólo lo podría hacer root.**

[https://es.wikipedia.org/wiki/Cron\\_\(Unix\)](https://es.wikipedia.org/wiki/Cron_(Unix))









**¡GRACIAS!**