

UNIX Y GOOGLE CLOUD

viernes 17 sep

COMANDOS UNIX

UI --- linux, mac

OS --- unix

CPU

En nombre de una distribución

alpine – solo lo basico

Abrimos en <https://bellard.org/jslinux/>

x86	Alpine Linux 3.12.0	Console	Yes	click here	url	
-----	---------------------	---------	-----	----------------------------	---------------------	--

Sale

```
Loading...

Welcome to JS/Linux (i586)

Use 'vlogin username' to connect to your account.
You can create a new account at https://vfsync.org/signup .
Use 'export file filename' to export a file to your computer.
Imported files are written to the home directory.

localhost:~#
```

Esto es el nombre del hostls

```
localhost:~#
```

En este caso es el ordenador local

Escribimos: ls y vemos el listado de opciones

ls mas asterisco: para buscar por extension

Si ponemos

```
localhost:~# ls -l hola
```

l es un argumento y hola un parámetro

```
localhost:~# ls -l
total 16
-rw-r--r--  1 root    root      114 Jul  5  2020 bench.py
-rw-r--r--  1 root    root       76 Jul  3  2020 hello.c
-rw-r--r--  1 root    root      22 Jun 26  2020 hello.js
-rw-r--r--  1 root    root     151 Jul  5  2020 readme.txt
localhost:~#
```

Primera fila los permisos. Si pone una d al principio es que es un directorio

4ª: tamaño fichero

5 a 7: dia mes año

8: nombre fichero

Si pone root es que tenemos los permisos, porque si ponemos whoami nos pone que somos root.

-t para ordenar por tiempo

-a para mostrar ocultos

Se puede poner ls -l -t -a o ls -lta indiferentemente

history para ver historial de comandos ejecutados en la sesion

Linux funciona con una jerarquia de carpetas. Para ver en cual estamos pwd. En este caso estamos en root. Podemos subir, bajar o saltar de carpeta.

```
localhost:~# pwd
/root
```

Para arriba cd ..

```
localhost:~# cd ..
localhost:/# pwd
/
```

Aquí estamos en la raíz. Al poner ls, estando en la raíz vemos las subcarpetas que hay en la raíz

bin: Se guardan los comandos que se pueden ejecutar.

Podemos hacer un listado de una carpeta, en este caso de bin con ls bin

```
localhost:/# ls bin
arch          false        lzop          rm
ash           fatattr      makemime      rmdir
base64        fdflush      mkdir         run-parts
bash          fgrep        mknod         sed
bbconfig      findmnt      mktemp        setpriv
bbsuid        fsync        more          setserial
busybox       getopt       mount         sh
busybox-extras grep         mountpoint    sleep
cat           gunzip       mpstat        stat
chgrp         gzip         mv            stty
chmod         hostname     netstat       su
chown         ionice       nice          sync
conspy        iostat      pidof         tar
cp            ipcalc       ping          touch
date          kbd_mode     ping6         true
dd            kill         pipe_progress umount
df            link         printenv      uname
dmesg         linux32      ps            usleep
dnsdomainname linux64       pwd           watch
dumpkmap      ln           rc-status     wdctl
echo          login        red           zcat
ed            ls           reformime     zsh
egrep         lsblk        rev           zsh-5.8
```

Cuando ejecutamos un comando, linux se va a la carpeta bin y ejecuta el comando que se encuentra ahí. Para saber de esa lista que es cada cosa ponemos ls -l bin. En este caso, vemos que al principio de cada fila no hay

En verde fichero, en azul carpeta.

Los archivos son nodo raíz, por lo que no hay nada debajo

Para bajar de raíz a la carpeta bin por ejemplo: cd bin

lib Donde se guardan las librerías

home Donde estas al arrancar

root Carpeta de usuario

Para ir de bin a root:

De bin a / y de / a root

O desde bin directo a root: cd /root. Este se puede usar en cualquier sitio, siempre que te sepas la ruta absoluta de la carpeta. Se llama saltar.

```
localhost:~# cd /root
localhost:~# ls
bench.py  hello.c  hello.js  readme.txt
localhost:~#
```

Vamos a root y vamos a:

Crear un fichero: touch name en name elegimos el nombre del fichero. Se suele poner la extensión también

```
localhost:~# ls -l
total 16
-rw-r--r--  1 root    root      114 Jul  5  2020 bench.py
-rw-r--r--  1 root    root       76 Jul  3  2020 hello.c
-rw-r--r--  1 root    root      22 Jun 26  2020 hello.js
-rw-r--r--  1 root    root     151 Jul  5  2020 readme.txt
localhost:~# touch prueba.txt
localhost:~# ls -l
total 16
-rw-r--r--  1 root    root      114 Jul  5  2020 bench.py
-rw-r--r--  1 root    root       76 Jul  3  2020 hello.c
-rw-r--r--  1 root    root      22 Jun 26  2020 hello.js
-rw-r--r--  1 root    root        0 Sep 17 15:52 prueba.txt
-rw-r--r--  1 root    root     151 Jul  5  2020 readme.txt
```

Para borrar el fichero: rm prueba.txt No hay comando para deshacer así que hay que estar seguro

```
localhost:~# ls -l
total 16
-rw-r--r--  1 root    root      114 Jul  5  2020 bench.py
-rw-r--r--  1 root    root       76 Jul  3  2020 hello.c
-rw-r--r--  1 root    root      22 Jun 26  2020 hello.js
-rw-r--r--  1 root    root        0 Sep 17 15:52 prueba.txt
-rw-r--r--  1 root    root     151 Jul  5  2020 readme.txt
localhost:~# rm prueba.txt
localhost:~# ls -l
total 16
-rw-r--r--  1 root    root      114 Jul  5  2020 bench.py
-rw-r--r--  1 root    root       76 Jul  3  2020 hello.c
-rw-r--r--  1 root    root      22 Jun 26  2020 hello.js
-rw-r--r--  1 root    root     151 Jul  5  2020 readme.txt
```

Crear una carpeta: mkdir name En name el nombre de la carpeta

```
localhost:~# ls -l
total 16
-rw-r--r--    1 root    root      114 Jul  5  2020 bench.py
-rw-r--r--    1 root    root       76 Jul  3  2020 hello.c
-rw-r--r--    1 root    root      22 Jun 26  2020 hello.js
-rw-r--r--    1 root    root     151 Jul  5  2020 readme.txt
localhost:~# mkdir carpetaprueba
localhost:~# ls -l
total 20
-rw-r--r--    1 root    root      114 Jul  5  2020 bench.py
drwxr-xr-x    2 root    root       37 Sep 17 15:55 carpetaprueba
-rw-r--r--    1 root    root       76 Jul  3  2020 hello.c
-rw-r--r--    1 root    root      22 Jun 26  2020 hello.js
-rw-r--r--    1 root    root     151 Jul  5  2020 readme.txt
```

Ahora para crear un archivo en la carpeta creada:

Vamos a la carpeta con `cd carpetaprueba` Una vez dentro de la carpeta, vacía como se ve usando ls, podemos crear un archivo con `touch hola.txt`

También se puede crear el archivo en la carpetaprueba, sin entrar, estando en root. Para ello, `touch carpetaprueba/prueba2.txt`

Si ahora vamos a la raíz con `cd /`, para crear otro archivo en la misma carpeta de antes, se debe en el comando touch especificar la ruta exacta de la carpeta `cd /root/carpetaprueba/prueba3.txt`

```
localhost:/# touch /root/carpetaprueba/prueba3.txt
localhost:/# cd /root/carpetaprueba
localhost:~/carpetaprueba# ls
hola.txt      prueba2.txt  prueba3.txt
```

Las flechas del teclado mueven por las últimas operaciones.

Al escribir una / y le damos al tabulador, muestra las opciones. Si solo hay una opción la rellena.

Para borrar la carpeta carpetaprueba: `rm -rf /root/carpetaprueba`. Este comando borra lo que hay dentro y luego la carpeta vacía

Ahora se pide en la carpeta root crear dos carpetas llamadas origen y destino, y en la primera un archivo llamado A.txt

```
localhost:(unknown)# cd /root
localhost:~# pwd
/root
localhost:~# mkdir origen
localhost:~# mkdir destino
localhost:~# touch origen/A.txt
localhost:~# ls
bench.py  destino  hello.c  hello.js  origen  readme.txt
localhost:~# cd origen
localhost:~/origen# ls
A.txt
```

Ahora vamos a pasar el archivo A.txt de la carpeta origen a destino. Hacemos `cp /root/origen/A.txt /root/destino/A.txt` Así se hace la ruta absoluta, también podemos hacerlo con rutas relativa cp, estando en este caso en origen: `cp A.txt ../destino.txt`

En vez de copiar se puede mover con `mv` en vez de `cp`. En el caso de mv, no es necesario indicar en destino el nombre del archivo: `cp /root/origen/A.txt /root/destino/` Si moviesemos el A.txt de origen a destino, habiendo ya uno, lo que hace es sobrescribir.

Estando en origen, podemos ver lo que hay en destino con `ls ../destino`

Para escribir en pantalla: `echo hola`

Para escribir en un archivo: `echo hola > A.txt`

Para ver lo que hay en el archivo `cat A.txt`

Otra opción es para muchas líneas, usar `tail` que muestra las últimas diez líneas.

Para abrir la ayuda de un comando, `ls --help`

En un comando colgado, CTRL + C cancela la ejecución

EJERCICIO

En root, crear hola.txt

```
localhost:~# cd /root
localhost:~# touch hola.txt
```

Escribir número en archivo

```
localhost:~# echo 123456789 > hola.txt
```

Crear directorio curso

```
localhost:~# mkdir curso
```

Mover hola.txt a curso

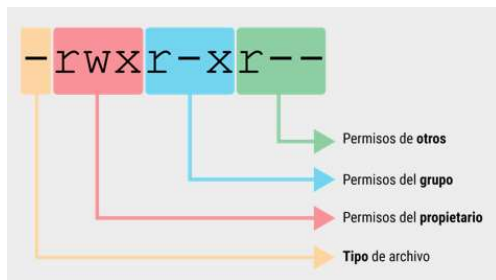
```
localhost:~# mv hola.txt curso/hola.txt
```

Listar directorios

```
localhost:~# ls -l
total 20
-rw-r--r--  1 root    root      114 Jul  5  2020 bench.py
drwxr-xr-x  2 root    root       62 Sep 17 16:33 curso
-rw-r--r--  1 root    root       76 Jul  3  2020 hello.c
-rw-r--r--  1 root    root       22 Jun 26  2020 hello.js
-rw-r--r--  1 root    root      151 Jul  5  2020 readme.txt

localhost:~# cd curso
localhost:~/curso# ls -l
total 4
-rw-r--r--  1 root    root       10 Sep 17 16:32 hola.txt
```


PERMISOS



Para dar permisos:

000 000 000

P G O

Para dar permisos, ponemos un uno en los permisos que queremos y luego lo pasamos a tres números decimales. Para dar todos los permisos a todos sería 777. Para eso se usa el comando chmod y un chmod calculator en internet para calcular el número que queremos.

Para el archivo `hola.txt` en la carpeta que estamos.

chmod 777 hola.txt

```
localhost:~/curso# ls -l
total 4
-rw-r--r--  1 root    root      10 Sep 17 16:32 hola.txt
localhost:~/curso# chmod 777 hola.txt
localhost:~/curso# ls -l
total 4
-rwxrwxrwx  1 root    root      10 Sep 17 16:32 hola.txt
```

Luego chgrp permite cambiar de grupo el archivo (se verá poco)

CUENTA GOOGLE CLOUD

Cloud ofrece: IAAS(infraestructura como servicio), PAAS(plataforma como servicio) y SAAS(software como servicio).


Comprobar que está todo parado al acabar de usar el cloud, para que no cobre de más

En caso de google cobra por el tiempo da igual que en ese tiempo lo usemos más o menos.


Nos registramos en Google Cloud Platform

En el menú izq: Compute engine -> Instancias de Vm.


Le damos a crear instancia

Nombre 
El nombre es permanente


instancia1

Etiquetas  (Opcional)

+ Agregar etiqueta

Región 
La región es permanente

europa-west2 (Londres)

Zona 
La zona es permanente

europa-west2-c

Configuración de la máquina

Familia de máquinas

Uso general Optimizada para procesamiento Memoria optimizada

Tipos de máquinas para cargas de trabajo comunes, optimizados en función del costo y la flexibilidad

Series


N1


Con la tecnología de la plataforma de CPU Intel Skylake o uno de sus predecesores


Tipo de máquina


f1-micro (1 CPU virtuales, 614 MB de memoria)

	vCPU	Memoria	GPU
	1 núcleo compartido	614 MB	-

Servicio de VM confidencial 
☐ Habilita el servicio de procesamiento confidencial en esta instancia de VM.


Contenedor 
☐ Implementa una imagen de contenedor en esta instancia de VM. [Más información](#)


Disco de arranque 


 Nuevo disco persistente equilibrado de 10 GB


Imagen

Debian GNU/Linux 10 (buster) Cambiar

Identidad y acceso a la API 

Cuenta de servicio 
Compute Engine default service account

Permiso de acceso 
☒ Permitir el acceso predeterminado
☐ Permitir el acceso total a todas las API de Cloud
☐ Configurar el acceso para cada API

Firewall 
Agregar etiquetas y reglas de firewall para permitir un tráfico de red determinado desde Internet

☐ Permitir tráfico HTTP
☐ Permitir tráfico HTTPS

⌵ Administración, seguridad, discos, redes, usuario único

Se usará tu crédito de la prueba gratuita para esta instancia de VM. [Nivel gratuito de GCP](#)

Crear Cancelar

Creamos la instancia, llamada instancia1

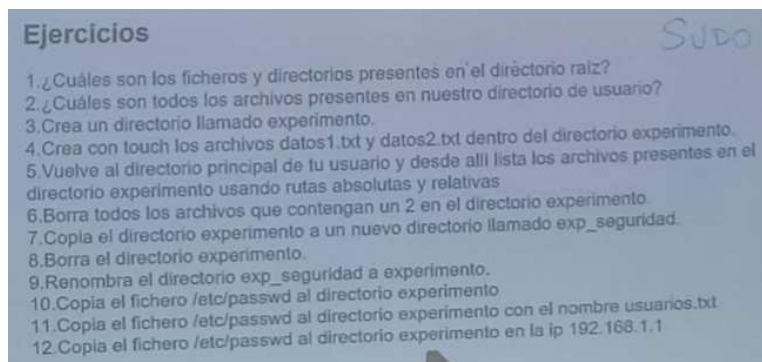
En opciones de SSH, abrimos en ventana de un navegador



Se nos abre una página de comandos. En este caso no somos administradores. Para serlo, escribiendo sudo su. Sin embargo, hay que usar la carpeta nuestra dentro de home. Para salir de root, usar comando exit

Si no sabemos todo el nombre de un archivo, podemos usar un * que funcionará como comodín. Borrar un dat* en una carpeta borrará todos los archivos que comienzan por dat en ese carpeta. Si se pone *3* borrará los archivos que contiene un 3.

EJERCICIOS



1.

```
sergi98lapunta@instancial:~$ cd /
sergi98lapunta@instancial:/$ ls
bin    dev    home   lib32  libx32  media  opt    root   sbin   sys    usr
boot   etc    lib    lib64  lost+found  mnt    proc   run    srv    tmp    var
```

2.

```
sergi98lapunta@instancial:/$ cd home
sergi98lapunta@instancial:/home$ cd sergi98lapunta/
sergi98lapunta@instancial:~$ pwd
/home/sergi98lapunta
sergi98lapunta@instancial:~$ ls
sergi98lapunta@instancial:~$
```

Nada

3.

```
root@instancia1:/home/sergi98lapunta# cd /root
root@instancia1:~# cd /home/sergi98lapunta/
root@instancia1:/home/sergi98lapunta# mkdir experimento
```

4.

```
root@instancia1:/home/sergi98lapunta# cd experimento
root@instancia1:/home/sergi98lapunta/experimento# pwd
/home/sergi98lapunta/experimento
root@instancia1:/home/sergi98lapunta/experimento# touch datos1.txt
root@instancia1:/home/sergi98lapunta/experimento# touch datos2.txt
```

5.

```
root@instancia1:/home/sergi98lapunta/experimento# cd /home/sergi98lapunta/
root@instancia1:/home/sergi98lapunta# ls -l
total 4
drwxr-xr-x 2 root root 4096 Sep 17 15:58 experimento
```

6.

```
root@instancia1:/home/sergi98lapunta# cd experimento
root@instancia1:/home/sergi98lapunta/experimento# ls
datos1.txt  datos2.txt
root@instancia1:/home/sergi98lapunta/experimento# rm *2*
root@instancia1:/home/sergi98lapunta/experimento# ls
datos1.txt
root@instancia1:/home/sergi98lapunta/experimento#
```

7.

```
root@instancia1:/home/sergi98lapunta# cp -r experimento exp_seguridad
root@instancia1:/home/sergi98lapunta# ls
exp_seguridad  experimento
```

8.

```
root@instancia1:/home/sergi98lapunta# rm -rf experimento
root@instancia1:/home/sergi98lapunta# ls
exp_seguridad
```

9.

```
root@instancia1:/home/sergi98lapunta# mv exp_seguridad experimento
root@instancia1:/home/sergi98lapunta# ls
experimento
```

10.

```
root@instancia1:/home/sergi98lapunta# cp /etc/passwd /home/sergi98lapunta/experimento/
```

11. (Poner el archivo como solo lectura)

```
root@instancia1:/home/sergi98lapunta/experimento# cp /etc/passwd /home/sergi98lapunta/experimento/usuarios.txt
root@instancia1:/home/sergi98lapunta/experimento# pwd
/home/sergi98lapunta/experimento
root@instancia1:/home/sergi98lapunta/experimento# ls
datos1.txt  passwd  usuarios.txt
```

```

root@instancia1:/home/sergi98lapunta/experimento# chmod 444 usuarios.txt
root@instancia1:/home/sergi98lapunta/experimento# ls -l
total 8
-rw-r--r-- 1 root root 0 Sep 17 16:08 datos1.txt
-rw-r--r-- 1 root root 1474 Sep 17 16:14 passwd
-r--r--r-- 1 root root 1474 Sep 17 16:15 usuarios.txt

```

Unidades de disco

df devuelve la ocupación de las distintas unidades de disco. Estamos en /dev/sda1. Cada unidad de disco o periféricos es una de las columnas. df -h para verlo en MB

```

root@instancia1:/home/sergi98lapunta/experimento# df
Filesystem      1K-blocks    Used Available Use% Mounted on
udev            288172         0    288172   0% /dev
tmpfs           59492      1804    57688   4% /run
/dev/sda1       10126600 1387892   8204592  15% /
tmpfs           297448         0    297448   0% /dev/shm
tmpfs           5120          0     5120   0% /run/lock
tmpfs           297448         0    297448   0% /sys/fs/cgroup
/dev/sda15      126710      5766    120944   5% /boot/efi
tmpfs           59488         0     59488   0% /run/user/1000
root@instancia1:/home/sergi98lapunta/experimento# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            282M   0    282M   0% /dev
tmpfs           59M   1.8M   57M   4% /run
/dev/sda1       9.7G  1.4G   7.9G  15% /
tmpfs           291M   0    291M   0% /dev/shm
tmpfs           5.0M   0     5.0M   0% /run/lock
tmpfs           291M   0    291M   0% /sys/fs/cgroup
/dev/sda15      124M   5.7M  119M   5% /boot/efi
tmpfs           59M   0     59M   0% /run/user/1000

```

ps permite ver los procesos activos

Para ejecutar un único comando como root, poner sudo al principio del comando.

Instalar apps:

apt update muestra la última versión de los paquetes que puedes necesitar. Si no deja, poner sudo al principio para dar permisos

Si queremos instalar algo que no está disponible, como java: sudo apt install java. Hay que hacer primero el comando de update. No funciona porque el nombre del paquete no es ese, hay que buscarlo en internet: El paquete es default-jdk, por lo que el comando es sudo apt install default-jdk.

Para saber donde está instalado, en este caso java: which java.

```

sergi98lapunta@instancia1:~$ which java
/usr/bin/java

```

Crear edem.txt en la carpeta nuestra:

Con vi edem.txt abrimos el fichero que permite escribir en el archivo

Escribimos lo que queremos, le damos a ESC y abajo escribimos

:w para guardar

:q para salir

:wq para las dos

En vez de vi otro método es con nano.

Una vez acabado, se borra la instancia.

Bash es un lenguaje que permite programar en el terminal.

Para probar se usa [Online Bash Shell - online editor \(onlinegdb.com\)](https://onlinegdb.com)

```
a=5
echo $a
```

```
5

...Program finished with exit code 0
Press ENTER to exit console.
```

Para ver si está instalado un programa: `git --version`. Si no devuelve nada no está.

EXERCICIO

Crear una máquina virtual, instalar git, crear un archivo script.h, con permisos de ejecución.

Añadir un código en el archivo:

```
sergi98lapunta@instance-1:~$ sudo apt-get install git
```

```
sergi98lapunta@instance-1:~$ pwd
/home/sergi98lapunta
sergi98lapunta@instance-1:~$ touch script.sh
sergi98lapunta@instance-1:~$ ls
script.sh
```

```
sergi98lapunta@instance-1:~$ vi script.sh
```

El código calcula $5*1 + 5*2 + 5*3 + \dots + 5*100$

```
#!/bin/bash
a=5
b=0
for i in `seq 1 100`;
do
    b=$(( $b + $(( $i * $a ))) )
done
echo $b
```

El archivo se ejecuta con el comando `sh`

```
sergi98lapunta@instance-1:~$ sh script.sh
25250
```