

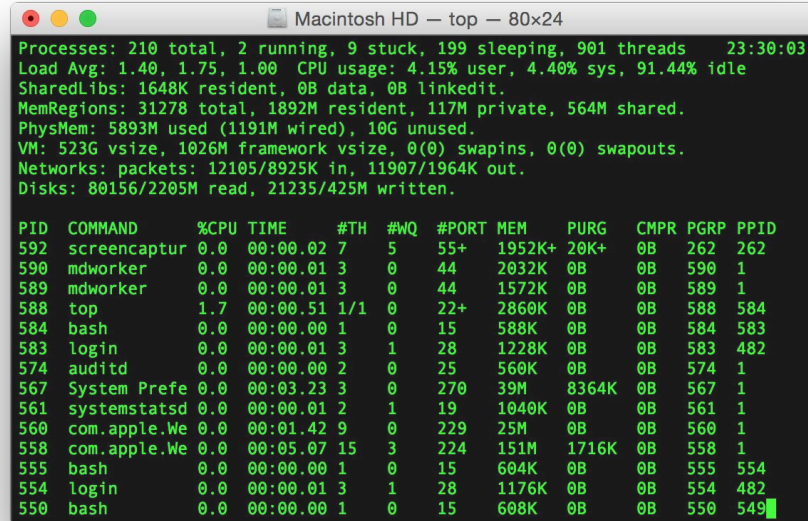
Sistemas Informáticos: Linux

1 DAM/DAW

REPASO

Modo Gráfico vs Modo no gráfico

Prompt → Terminal



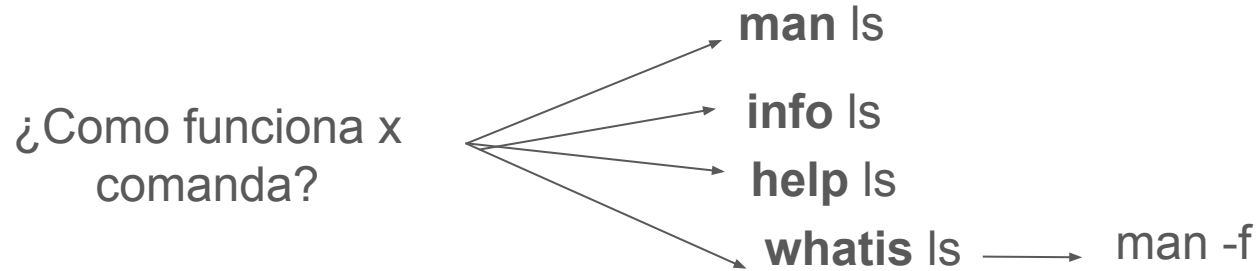
```
Macintosh HD — top — 80x24
Processes: 210 total, 2 running, 9 stuck, 199 sleeping, 901 threads 23:30:03
Load Avg: 1.40, 1.75, 1.00 CPU usage: 4.15% user, 4.40% sys, 91.44% idle
SharedLibs: 1648K resident, 0B data, 0B linkedit.
MemRegions: 31278 total, 1892M resident, 117M private, 564M shared.
PhysMem: 5893M used (1191M wired), 10G unused.
VM: 523G vsize, 1026M framework vsize, 0(0) swapins, 0(0) swapouts.
Networks: packets: 12105/8925K in, 11907/1964K out.
Disks: 80156/2205M read, 21235/425M written.

PID  COMMAND  %CPU  TIME    #TH   #WQ   #PORT  MEM    PURG    CMPR  PGRP  PPID
592  screencaptur  0.0  00:00.02  7     5    55+   1952K+ 20K+   0B    262  262
590  mdworker    0.0  00:00.01  3     0    44    2032K  0B     0B    590  1
589  mdworker    0.0  00:00.01  3     0    44    1572K  0B     0B    589  1
588  top         1.7  00:00.51  1/1   0    22+   2860K  0B     0B    588  584
584  bash        0.0  00:00.00  1     0    15    588K   0B     0B    584  583
583  login       0.0  00:00.01  3     1    28    1228K  0B     0B    583  482
574  auditd      0.0  00:00.00  2     0    25    560K   0B     0B    574  1
567  System Prefe 0.0  00:03.23  3     0    270   39M    8364K  0B     0B    567  1
561  systemstatsd 0.0  00:00.01  2     1    19    1040K  0B     0B    561  1
560  com.apple.We 0.0  00:01.42  9     0    229   25M    0B     0B    560  1
558  com.apple.We 0.0  00:05.07  15    3    224   151M   1716K  0B     0B    558  1
555  bash        0.0  00:00.00  1     0    15    604K   0B     0B    555  554
554  login       0.0  00:00.01  3     1    28    1176K  0B     0B    554  482
550  bash        0.0  00:00.00  1     0    15    608K   0B     0B    550  549
```

REPASO

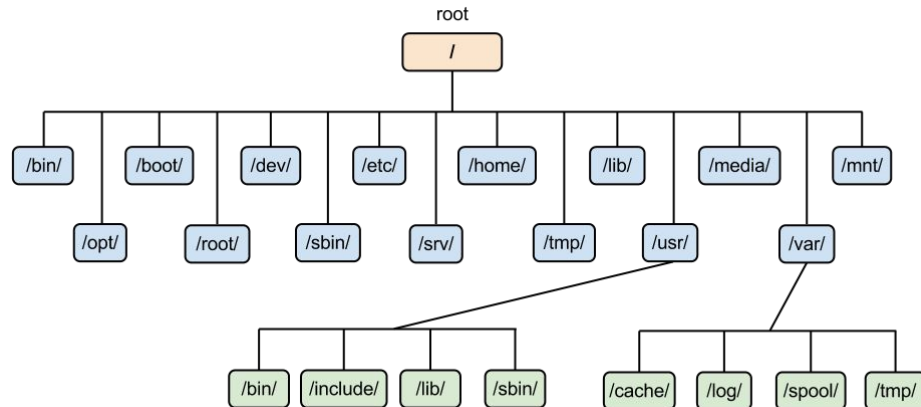
ls	List directory contents
ls -l	List directory contents + long list format
ls -la	List directory contents + long list format +all (sin ignorar archivos ocultos)
cd	Cambiar de directorio
pwd	Mostrar directorio actual
clear	Borrar la pantalla de la terminal

REPASO



NAME	Nombre del comando y una breve descripción.
SYNOPSIS	Proporciona ejemplos de cómo se ejecuta el comando
DESCRIPTION	Proporciona una descripción más detallada del comando.
SEE ALSO	Proporciona una idea de dónde puedes encontrar información adicional. También suele incluir otros comandos que están relacionados con este comando.

Ruta Absoluta	C:\Usuarios\Usuario\Documentos\doc.txt
Ruta Relativa	Documentos\doc.txt
cp	Copiar
mv	Mover
rename	Rename multiple files
cp -i, mv -i // cp -v, mv -v	Pregunta usuario / muestra esquema



REPASO

\$: touch file.txt

\$: echo "Escribiendo en un archivo file" > file.txt

\$: echo "Sobreescribiendo en archivo file" >> file.txt



REPASO

Variables de entorno vs variables locales

\$: var=50

\$: echo \$var
→ 50

\$: echo **\$PATH**

\$: **export** var

\$: **env** | **grep** var

\$: **unset** var

REPASO

¿Alias?

‘Comillas simples’ vs “Comillas dobles”

¿Permisos?

touch	Crear fichero
cat	Mostrar fichero
rm // rm -i	Eliminar fichero
rmdir // rmdir -i // rmdir -r	Eliminar fichero
mkdir	Crear directorio

7.2 Comprimir Archivos

Comprimir archivos los hace más pequeños eliminando la duplicación en un archivo y guardándolo de tal manera que el archivo se pueda restaurar. Un archivo de texto legible podría **reemplazar palabras usadas con frecuencia por algo más pequeño**, o una imagen con un fondo sólido podría representar manchas de ese color por un código.

- **Lossless** (o «sin pérdida»)
- **Lossy** (o «con pérdida»)



gzip ; gunzip
¿gzip -l?

bzip2 ; bunzip2
¿bzip2 -v?

zip ; unzip
(menos frecuente en Linux)



7.3 Empaquetar Archivos



tar -cf files.tar file1.txt file2.txt
Create tar (.tar) file

tar -czf files.tgz file1.txt file2.txt
Create gzip tar (.tgz) file

Extraer (tgz): **tar -xzf** access_logs.tgz
Extraer (tbz): **tar -xjf** access_logs.tbz



- Para extraer utilizar la marca **-x**
- Mientras que UNIX no trata las extensiones de archivo especialmente, la convención es usar .tar para los archivos tar y .tar.gz o .tgz para los archivos tar comprimidos con gzip.
- Se puede utilizar bzip2 en vez de gzip sustituyendo la letra **z** con una **j** y usando .tar.bz2,.tbz, o .tbz2 para una extensión de archivo.

7.3 Empaquetar Archivos



tar -cf files.tar file1.txt file2.txt
Create tar (.tar) file

tar -czf files.tgz file1.txt file2.txt
Create gzip tar (.tgz) file

Extraer (tgz): **tar -xzf** access_logs.tgz

Extraer (tbz): **tar -xjf** access_logs.tbz



- Para añadir un archivo a un archivo empaquetado existente, utiliza la opción **-r** con el comando tar:

tar -rf files.tar file_to_add.txt

Capítulo 8

head, head -x	Mostrar las primeras líneas de un archivo
tail, tail -x	Mostrar las últimas líneas de un archivo
nl	Enumerar las entradas
wc, wc -l, wc -w, wc -c	Mostrar líneas, palabras y bytes de un archivo
sort, sort -n, sort -r	Ordenar

STDOUT, STDERR, STDIN

Capítulo 8

Salida estándar o **STDOUT** es la salida normal de los comandos. Cuando un comando funciona correctamente (sin errores), la salida que produce se llama STDOUT.

Error estándar o **STDERR** son mensajes de error generados por los comandos. De forma predeterminada, STDERR se muestra en la ventana de la terminal (pantalla) donde se ejecuta el comando. Esto se puede cambiar para que se redirija a un archivo.

La entrada estándar **STDIN** es la información normalmente introducida por el usuario mediante el teclado.

Redireccionando STDOUT	8.3.4: echo "Line 1" > example.txt
Redireccionando STERR	8.3.5: ls /fake 2> error.txt
Redireccionando Múltiples Secuencias	8.3.6: ls /fake /etc/ppp > example.txt 2> error.txt
Redireccionando STDIN	8.3.7: tr 'a-z' 'A-Z' < example.txt > newexample.txt
STDOUT y STDERR	8.3.6:ls /fake /etc/ppp &> error.txt

Capítulo 8

El comando **find** es una herramienta muy poderosa que puedes utilizar para buscar archivos en el sistema. Este comando puede buscar archivos por nombre, incluso usando los caracteres comodín cuando no estás seguro del nombre exacto del archivo. Además, puedes buscar los archivos en función de otros parámetros, tales como tipo de archivo, tamaño de archivo, propiedades del archivo

find [directororio de inicio] [opción de búsqueda] [criterio de búsqueda]

- / (**slash**) – busca en todo el sistema.
- . (**punto**) – busca en la carpeta en la que estás trabajando actualmente (directorio actual).
- ~ (**tilde**) – para buscar desde tu directorio home.



Capítulo 8

find [directorio de inicio] [opción de búsqueda] [criterio de búsqueda]

```
$: find . -name “*bash*”
```

```
$: find . -mmin -5
```

```
$: find . -size +2M
```

```
$: find . -mtime 0
```

```
$: find . -mtime 1 #archivos  
modificados hace más de 1día.
```

```
$: find . -mtime -1 #archivos  
modificados hace más de 1día.
```



Capítulo 8

\$: more file.txt

\$: less file.txt

Espacio	Hacia delante
b	Hacia atrás
q	Salir



“Manera rápida de visualizar ficheros largos”

Capítulo 9:

Scripting Básico

```
#!/bin/bash
```

```
~root: env X="() { :;} ; echo shellshock" /bin/sh -c "echo completed"
```

```
> shellshock
```

```
> completed
```

Capítulo 9: ¿Qué es un script shell?

Un **script shell** es un archivo de comandos ejecutables que ha sido almacenado en un archivo de texto. Cuando el archivo se ejecuta, se ejecuta cada comando. Los scripts shell tienen acceso a todos los comandos del shell, incluyendo la lógica. Un script (o «secuencia de comandos» en español), por lo tanto, puede detectar la presencia de un archivo o buscar una salida particular y cambiar su comportamiento en consecuencia. Se pueden crear scripts para automatizar partes repetitivas de tu trabajo, que ahorran tu tiempo y aseguran consistencia cada vez que utilices el script.



Capítulo 9

Script ha de empezar con la línea: **#!/bin/bash**

Extensión: **.sh**

Ejecución: **./scrip_file.sh**

¿No tengo permisos?: **chmod u+x script_file.sh**

Capítulo 9

UNIX tiene muchos editores de texto y las ventajas del uno sobre el otro se debaten muy a menudo. Los principales editores de Texto: **vi (vim)**, **gedit**, **nano...**

¡Probémoslos!



Capítulo 9: Las variables

Las variables son una parte esencial de cualquier lenguaje de programación.

```
#!/bin/bash
```

```
ANIMAL="penguin"
```

```
echo "My favorite animal is a $ANIMAL"
```

```
#!/bin/bash
```

```
CURRENT_DIRECTORY=`pwd`
```

```
echo "You are in $CURRENT_DIRECTORY"
```

Capítulo 9: Condicionales

```
if sometCommand
then
    # do this if some command has an exit code of 0
else
    #.....
fi
```

```
if [ condition ]
then
    # Condition is true statement
else
    # Condition is not true
fi
```

Capítulo 9: Condicionales

```
#!/bin/bash
```

```
a=99
```

```
b=45
```

```
if [ $a -lt $b ]
```

```
then
```

```
    echo "a is less than b"
```

```
else
```

```
    echo "a is greater than b"
```

```
fi
```

-eq	==, =
-ne	!=
-gt	>
-ge	>=
-lt	<
-le	<=



No admiten operador [], sí (())

Capítulo 9: Condicionales



¡Ojo con los **ESPACIOS!**

Capítulo 9: Los Loops

Los loops (o «ciclos o bucles» en español) permiten que un código se ejecute repetidas veces. Pueden ser útiles en numerosas situaciones, como cuando quieres ejecutar los mismos comandos sobre cada archivo en un directorio, o repetir alguna acción 100 veces. Hay dos loops principales en los scripts del shell: el loop **for** y el loop **while**.

```
for i in {1..1000}
do
    echo $i
done
```

Más info: 9.4.3

Capítulo 9: read

El comando read puede aceptar una cadena directo desde el teclado o como parte de la redirección de comandos tal como aprendiste en el capítulo anterior.

```
#!/bin/bash
```

```
read user_text
```

```
echo "El usuario ha introducido: $user_text"
```

Capítulo 9: Case

read **x**

case **\$x** in

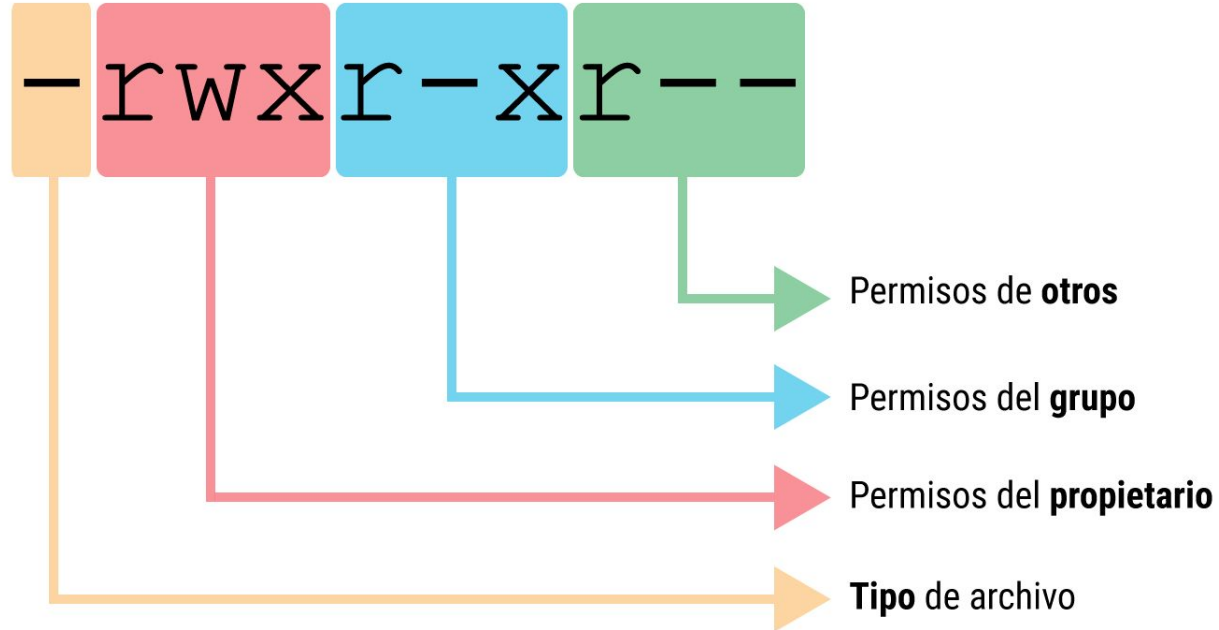
1)echo "uno";;

2)echo "dos";;

***)**echo "no sé qué numero es";;

esac

Ojo: Permisos de linux



Capítulo 9: Problema 1. a)

Crea un script que cree **100 archivos llamados filex.txt** en el directorio actual. A continuación, **muestre el contenido del directorio** y le **pregunte al usuario si desea borrar los archivos generados (Y/N)**. Si el usuario quiere borrarlos presiona (Y) y los archivos se borrarán apareciendo el mensaje *“Archivos generados borrados correctamente”*. En cambio, si el usuario decide no borrarlos (N), los archivos se conservarán y se imprimirá por pantalla el mensaje *“Archivos creados y conservados”*.

Capítulo 9: Problema 1. b)

Crea un script que cree **100 archivos llamados filex.txt** en el directorio actual. A continuación, **muestre el contenido del directorio** y le **pregunte al usuario si desea borrar los archivos generados (Y/N)**. Si el usuario quiere borrarlos presiona (Y) y los archivos se borrarán apareciendo el mensaje *“Archivos generados borrados correctamente”*. En cambio, si el usuario decide no borrarlos (N), los archivos se conservarán y se imprimirá por pantalla el mensaje *“Archivos creados y conservados”*.

Cuando se creen los archivos, añada permisos de lectura, escritura y ejecución al usuario principal. A grupos y otros usuarios quita todos los permisos.

Capítulo 9: Problema 2

Parte 1: Utilizando el contenido del archivo 'netflix2.csv' (enlace en classroom), el programa deberá organizar alfabéticamente las películas, eliminar cualquier información adicional y evitar información duplicada. Guardará el nuevo archivo “limpio” en otro archivo llamado 'netflix.csv'.

Parte 2: Después, el usuario podrá seleccionar una de las siguientes opciones:

NETFLIX

Menú de selección de películas

1.Recomendación rápida

2.Listar por año

Ingrese el número de la opción deseada

Capítulo 9: Problema 2

Parte 1:

1. Eliminar la primera línea del archivo “netflix2.csv”.
2. Ordenar alfabéticamente las películas listadas en “netflix2.csv”.
3. Eliminar cualquier película duplicada.
4. Guardar el resultado en un archivo llamado “netflix.csv”.

Parte 2: Después, el usuario podrá seleccionar una de las siguientes opciones:

NETFLIX

Menú de selección de películas

- 1.Recomendación rápida → *select_random_movies()*
- 2.Listar por año → *select_movies_by_year()*

Ingrese el número de la opción deseada

Capítulo 9: Problema 2

select_random_movies():

Elegir película de la carta aleatoriamente.

select_movies_by_year():

Pedir al usuario que introduzca un año: *“Ingrese el año de la publicación”*

Comprobar que el año introducido está en el rango de 1950-2024.

Si año dentro del rango muestra por pantalla al usuario una lista con las películas publicadas en el año introducido por el usuario.

Si el año no está dentro del rango muestra: *“Año no válido. Cerrando el programa”*

Capítulo 9: Problema 3

- Ubicar archivo1.txt archivo2.txt archivo3.txt archivos en el directorio actual.
- Crear script llamado *p3.sh*
- Desde el script, crear un directorio llamado *archivosPractica*.
- Mover *archivo1.txt*, *archivo2.txt* y *archivo3.txt* al directorio *archivosPractica*.

Capítulo 9: Problema 3

- Movernos a *archivosPractica*
- Utiliza un bucle for que elimine todos los permisos de cada archivo, para luego otorgar permisos de lectura y escritura al propietario y solo permisos de lectura a los demás usuarios para los archivos *archivo1.txt*, *archivo2.txt* y *archivo3.txt*.
- Mover *archivo3.txt* al directorio *archivosPractica/respaldo*.
- Utiliza el comando *gzip* para comprimir *archivo1.txt* dentro del directorio *archivosPractica*

Capítulo 9: Problema 3

- Utiliza el comando head para mostrar las primeras 5 líneas del archivo *archivo2.txt* y redirige la salida al archivo *primeras_lineas.txt*.
- Utiliza el comando tail para mostrar las últimas 5 líneas del archivo *archivo2.txt* y redirige la salida al archivo *ultimas_lineas.txt*.