SISTEMAS INFORMÁTICOS

COMANDOS LINUX

PWD

- Sintaxis: pwd
- Viene de "Print Work Directory" y sirve para saber el directorio donde estamos situados
- El prompt suele mostrar esa información, pero utiliza el símbolo "~" cuando nos encontramos en nuestro directorio personal
- Ejemplos

```
pwd
```

WHOAMI

- Sintaxis: whoami
- Muestra el usuario de la sesión
- Ejempos

```
whoami
```

HOSTNAME

- Sintaxis:

hostname

- Muestra el nombre del equipo
- Ejempos

```
hostname
```

CD

- Sintaxis: cd [directorio]
- (Change directory) Permite cambiar de directorio
- Ejemplo:

```
cd /home
cd .. # Para ir al directorio padre
cd /tmp/
cd ~ # Para ir al directorio personal
```

HISTORY

- Sintaxis: history [opcion ...]
- Muestra el historial de comandos introducidos por el usuario
- El fichero `.bash_history` contiene el listado de todos los comandos ejecutados en sesiones anteriores, este fichero se encuentra en cada uno de los directorios personales de los usuarios
- Podemos mostrar el contenido del fichero con

```
cat .bash_history
```

- Cuando nosotros cerramos la sesión se almacenan en este fichero todos los comandos ejecutados durante la sesión
- El comando `history` muestra la información del fichero `.bash_history` y los comandos ejecutados durante la sesión, todos los comandos mostrados van precedidos de un número. Podemos ejecutar el comando del historial mediante ese número como se muestra a continuación

```
!5 # Ejecuta el comando número 5 del listado
```

- Opciones:
    - -c => Borra el historial
    - -w => Guarda el historial de comandos de sesión en el fichero
- Ejemplos

```
history
history -c # Borra el historial
history -w # Guarda el historial de la sesión
```

UNAME

- Sintaxis: uname [opcion ...]
- Muestra la información sobre nuestro linux
- Opciones:
    - -a => Muestra toda la información del sistema
    - -r => Muestra la versión de nuestro núcleo linux
    - -n => Muestra el nombre de la máquina
    - -m => Muestra la arquitectura de la máquina
- Ejemplos:

```
uname
uname -r # Versión del núcleo
uname -a # Toda la información
uname -n # Nombre de la máquina
uname -m # Arquitectura de la máquina
```

MAN

- Muestra la ayuda de comandos y ficheros:
    - Muestra el propósito de comando
    - Sintaxis
    - Opciones
- Sintaxis: man [seccion] termino
- Las páginas del man están almacendas en un directorio `/usr/share/man`
- Cuando instalamos un programa instala también las páginas de ayuda dentro de `/usr/share/man/manX`, donde la X corresponde a las distantes secciones del man
- El término buscado puede ser:
    - Comandos. De cada comando muestra:
        - Propósito del comando
        - Sintaxis
        - Opciones
    - Ficheros (Mostrará la estructura del fichero)
    - Ficheros especiales.

- Secciones:
    - a. Comandos generales Ej: ls, cd, ...
    - b. Llamadas al sistema
    - c. Biblioteca de funciones
    - d. Ficheros especiales Ej: /dev/null
    - e. Ficheros de configuración Ej: /etc/passwd
    - f. Juegos y salvapantallas
    - g. Miscelania
    - h. Comandos de administración del sistema (Trata como crear particiones en disco, reinicios, ...)
- Ejemplos:

```
man ls
man 5 passwd # Mostraría información sobre el fichero passwd
man passwd # Mostraría información sobre el comando passwd
```

CAT

- Muestra el contenido por pantalla del fichero
- Sintaxis: cat [opciones] fichero [ficheros]
- Opciones:
    - -n => Muestra los números de línea
    - -b => Numerá las líneas menos las vacías
- Ejemplos:

```
cat numeros.txt
cat -n numeros.txt
cat -b numeros.txt
```

- El comando `tac` muestra el contenido del fichero al revés

NL

- Sintaxis: nl [opciones] fichero [ficheros]
- Numera las líneas, es exactamente igual que `cat -n`
- Opciones:
    - -ba -> Numera las líneas en blanco
- Ejemplos:

```
seq 1 10 > numeros.txt # Para generar el fichero
nl numeros.txt # Numéra las líneas menos líneas en blanco
nl -ba numeros.txt # Numera todo
```

HEAD

- Muestra las líneas iniciales del fichero (Por defecto 10 líneas)
- Sintaxis: head [opciones] fichero [ficheros]
- Opciones:
    - -n => Indica el número de líneas a mostrar
- Ejemplos:

```
head numeros.txt
head -n 10 nombres.txt
```

TAIL

- Sintaxis: tail [opciones] fichero [ficheros]
- Muestra las líneas FINALES del fichero
- Opciones:
    - -n => Indica el número de líneas a mostrar
    - -f => No cierra el fichero y espera a mostrar las núevas líneas introducidas
- Ejemplos

```
tail -n 10 numeros.txt
tail -f numeros.txt
```

WC

- Cuenta líneas, palabras y caractéres
- Sintaxis: wc [opciones ...] fichero [ficheros]
- Opciones:
    - -l => Muestra solamente el número de líneas
    - -w => Muestra solamente el número de palabras
    - -m => Muestra solamente el número de caractéres
- Ejemplos:

```
wc numeros.txt
wc -l numeros.txt
wc -wnumeros.txt
wc -m numeros.txt
```

SORT

- Sintaxis: sort [opciones...] fichero [ficheros]
- Ordena las líneas del fichero
- Si especificamos más de un fichero ordena todo, no cada fichero por separado
- Opciones:
    - -k numero => Número de columna a cordenar
    - -t separador => Indica el separador de columnas
    - -n => Realiza la ordenación númerica (por defecto es alfabética)
    - -r => Ordena en orden inverso
    - -f => Ignora entre mayúsculas y minúsculas
- Ejemplos:

```
sort -r nombres.txt
sort -k 2 -t' ' nombres.txt
sort -n nombres.txt
```

UNIQ

- Sintaxis: uniq [opciones] fichero
- Muesta el contenido de un fichero quitando las líneas repetidas.
- Para quitar las líneas repetidas dichas líneas deben estar contiguas (podemos apoyarnos en el comando `sort` para esta labor)
- Opciones:
    - -c => Muestra el número de repeticiones
    - -d => Muestra solo las líneas repetidas
    - -u => Muestra solo las líneas no repetidas
- Ejemplos:

```
uniq numeros.txt # No quita las repeticiones discontinuas
sort numeros.txt | uniq # Quita las repeticiones aún siendo discontinuas
uniq -c numeros.txt
```

CUT

- Sintaxis: cut [opciones] fichero
- Cut extrae columnas de un fichero,es decir, corta un fichero por columnas
- Opciones:
    - -f numeros => Determina las columnas que quiero extraer, siendo números los números de columna separados por comas
    - -d separador => Indica el separador de columnas
    - -c numeros => Determina los caracteres o secuencia de caracteres a extraer
- Ejemplos:

```
cut -f2,3 -d':' nombres.txt
cut -f2-5 -d':' nombres.txt
cut -c 5-10 nombres.txt
```

SPLIT

- Sintaxis: split [opciones] fichero [prefijo]
- Divide un fichero en ficheros más pequeños, enta división puede ser por líneas o por tamaño
- El comando split nombrará los ficheros resultado de la división como xaa, xab, xac, ...
- Se puede indicar un prefijo con el que nombrará a los ficheros
- Opciones:
    - -l numero => Indica el número de líneas por las que realizará la división
    - -b tamaño => Indica el tamaño (B,K,M,G, ...) por el que realizará la división
- Ejemplos:

```
split -l 20 numeros.txt # Cada fichero resultado de la división tendrá 20
líneas
split -b 10K nombres.txt # Cada fichero resultado de la división tendrá 10K
split -l 10 numeros.txt num # Los ficheros se nombrarán como numaa, numab,
numac, ...
```

- Se pueden unir los ficheros mediante el comando `cat`

```
cat x*
```

COMANDO JOIN

- Sintaxis: join [opciones] fichero1 fichero2
- Une dos ficheros con un campo en común
- Opciones
    - -t => Delimitador de los campos del fichero
    - -1 num => Posición del campo en común en el primer fichero
    - -2 num => Posición del campo en común en el segundo fichero
- Ejemplos:

```
$ cat nombresyprimerapellido.txt
german carreño
alejandro castaño
victor xavier

$ cat nombresysegundoapellido.txt
german hevia
alejandro fonseca
victor rodriguez

$ join -t' ' -1 1 -2 1 nombresyprimerapellido.txt nombresysegundoapellido.txt
german carreño hevia
alejandro castaño fonseca
victor xavier rodriguez
```

GREP

- Sintaxis: grep patron [ficheros]
- Muestra las líneas en donde se encuentra el patrón
- Opciones:
    - -i => No distingue entre mayúsculas y minúsculas
    - -c => Cuenta el número de líneas donde aparece el patrón
    - -e patron => Podemos especificar varios patrones
    - -v => Muestra las líneas donde NO encuentra el patrón
    - -A numero => Muestra el número de líneas indicado después de la coincidencia del patrón

    - -B numero => Muestra el número de líneas indicado antes de la coincidencia del patrón
    - -H => Incluye el nombre del fichero en el resultado de la búsqueda
    - -h => Quita el nombre del fichero en el resultado de la búsqueda

- Ejemplos:

```
# Trabajaremos sobre este fichero
$ cat nombres.txt
1 german carreño hevia
2 alejandro castaño fonseca
3 victor xavier rodriguez

# Encuentra germán en la primera línea y la muestra
$ grep "german" nombres.txt
1 german carreño hevia

# No encuentra nada porque distingue entre mayúsculas y minúsculas
$ grep "German" nombres.txt

# Encuentra porque -i permite no distinguir entre mayúsculas y minúsculas
$ grep -i "German" nombres.txt
1 german carreño hevia

# Muestra el numero de líneas donde encuentra el patrón "ca"
$ grep -c "ca" nombres.txt
2

# Muestra las líneas donde NO encuetra el patrón
$ grep -v "german" nombres.txt
2 alejandro castaño fonseca
3 victor xavier rodriguez

# Busca recursivamente el patrón
$ grep -R "german" *
nombres.txt:1 german carreño hevia
nombresyprimerapellido.txt:german carreño
nombresysegundoapellido.txt:german hevia
numeros.txt:1 german carreño hevia
```

EXPRESIONES REGULARES

EXPRESIONES REGULARES BÁSICAS

- "^" => Indica el comienzo de línea
- "$" => Indica el final de línea
- "." => Sustituye a cualquier carácter
- "[caracteres]" => Permite buscar un carácter de entre un conjunto
- Ejemplos:

```
# Líneas que comienzan con un 1
$ grep "^1" nombres.txt
1 german carreño hevia

# Líneas que acaban por "a" o "A"
$ grep -i "a$" nombres.txt
1 german carreño hevia
2 alejandro castaño fonseca

# Líneas que solo tengan 3 caracteres cualesquiera
$ grep "^...$" nombres.txt

# Líneas que comiencen con 1 o 2
$ grep -i ^[12] nombres.txt
1 german carreño hevia
2 alejandro castaño fonseca

# Podemos negar dentro de la agrupación
# Líneas que no comiencen por 1 o 2
grep -i ^[^12] nombres.txt
3 victor xavier rodriguez
```

EXPRESIONES REGULARES EXTENDIDAS

- Se ejecutan mediante `grep -E` o `egrep`
- Solo tienen efecto a la expresión que tienen a su izquierda
- Expresiones de repetición:
    - "?" => Cero o una vez
    - "*" => Cero, una o muchas veces
    - "+" => Una o muchas veces
    - "{n}" => n veces
    - "{n, m}" => Entre n y m veces
    - "{n,}" => Como mínimo n veces
- Otras:
    - "()" => Agrupación
    - "|" => Alternativa
- Ejemplos:

```
# Trabajaremos con este fichero
cat numeros.txt | head
1
2
...
999
1000

# Encuentra números de 1 o 2 dígitos
egrep "^[0-9][0-9]?$" numeros.txt
1
2
...
99
1000

# Números que empiezen por 5
egrep "^5[0-9]*$" numeros.txt
5
50
53
...
59
500
501
...

# Números con unidades o decenas
egrep "^[0-9][01]+$" numeros.txt
10
11
```

```
20
21
30
31
40
41
...

# Unidades
egrep "^[0-9]{1}$" numeros.txt
1
...
9

# Unidades y decenas
egrep "^[0-9]{1,2}$" numeros.txt
1
...
9

# decenas, centenas, ...
egrep "^[0-9]{2,}$" numeros.txt
1
...
1000
```

TR

- Sintaxis: flujo | tr [opciones] [conjunto1] [conjunto2]
- Sustituye los caracteres del conjunto1 por los del conjunto2 en el orden en que se los encuentra
- Opciones
    - -s conjunto => Elimina todos los duplicados (deben estar contiguos)
    - -d conjunto => Elimina caracteres
- Ejemplos:

```
# Trabajaremos sobre este fichero
$ cat nombres.txt
1 german carreño hevia
2 alejandro castaño fonseca
3 victor xavier rodriguez

# Sustituye mayúsculas por minúsculas
$ cat nombres.txt | tr [a-zñ] [A-ZÑ]
1 GERMAN CARREÑO HEVIA
2 ALEJANDRO CASTAÑO FONSECA
3 VICTOR XAVIER RODRIGUEZ

# Elimina los duplicados
$ echo "Germaaaan" | tr -s "a"
```

```
German

# Elimina el caracter a y n
$ echo "Germaaaan" | tr -d "an"
Germ
```

ECHO

- Sintaxis: echo [opciones] argumentos
- Este comando muestra los argumentos en la salida estandar
- Podemos mostrar variables de entorno, estas variables se especifican con un símbolo "$" delante, por ejemplo, `$USER`
- Existen variables de entorno que se crean cuando entras en sesión
    - $USER
    - $PWD
    - $OLDPWD
    - $HOME
- Podemos definir nuestras propias variables de entorno
- Ejemplos:

```
# Declaramos y asignamos valor a la variable DIR
$ DIR="/etc"
# Utilizamos la variable DIR para mostrar el contenido del directorio /etc
mediante la variable $DIR
$ ls -ld $DIR
```

- Cuando mostramos por pantalla cadenas de caracteres mediante el comando `echo`, podemos utilizar comillas simples o dobles
    - Las comillas dobles, permiten sustituir el valor de la variable dentro de la cadena
    - Las comillas simples, no sustituyen el valor de la variable dentro de la cadena
- Ejemplos:

```
# Declaramos y asignamos valor a la variable nombre
nombre="Pepe"

echo "Hola $nombre" # Muestra Hola Pepe
echo 'Hola $nombre' # Muestra Hola $nombre
```

- Opciones:
    - -n => Incluye salto de línea

    - -e => Permite utilizar caractéres especiales \n, \t, ...

- Ejemplos:

```
echo -n "Hola mundo" # Imprime Hola mundo y salta a la sínea siguiente
echo -e "\t\t\tHola mundo\n" # Imprime Hola mundo con tabulaciones al
principio y salto de línea al final
```

ENV

- Sintaxis: env
- Este comando muestra todas las variables de entorno

```
# Muestra por pantalla todas las variables de entorno
$ env
```

LS

- Sintaxis: ls [opciones] [ficheros/patron]
- Listar el contenido de uno o varios directorios
- Si no introducimos un fichero/s o patron/es muestra el contenido del directorio actual
- En debian el comando `ls` es un alias de `ls --color=auto`, si queremos ver el alias `alias ls`
- Opciones
    - -a => Muestra los archivos ocultos / o que comienzan por punto
    - -l => Formato largo (Permisos, número de subdirectorios/enlaces, usuario propietario, grupo, tamaño, fecha de modificación, nombre)
    - -h => Muestra el formato leible para el usuario la cantidad de bytes
    - -R => Lista recursivo
    - -t => Lista ordenadamente por fecha de modificación
    - -S => Lista ordenadamente por tamaño
    - -r => Invierte el orden
    - -1 => Muestra cada fichero/directorio/enlace en una línea

Ejemplos:

```
# Muestra el contenido del directorio actual
$ ls

# Muestra el contenido del directorio actual con los ficheros ocultos y en
formato largo
$ ls -al

# Muestra los ficheros acabados en .avi ordenados por tamaño
$ ls -S *.avi

# Muestra el contenido del directorio /etc y el de sus subdirectorios ...
$ ls -R /etc
```

MKDIR

- Sintaxis: mkdir [opciones] directorio [directorios]
- Crea directorio o directorios
- Opciones
    - -m permisos
    - -p => Crea directorios anidados
    - -v => Muestra una traza de las operaciones
- Ejemplos:

```
# Crea el directorio "test"
mkdir test

# Crea el directorio test con permisos de rwx solamente para el usuario
propietario
mkdir -m 700 test

# Error si los directorios test y sdk no existen
mkdir test/sdk/java

# Crea todos los directorios anidados
mkdir -p test/sdk/java
```

CP

- Sintaxis: cp [opciones] origen [origenes] destino
- Copia ficheros y directorios en un destino
- Se pueden especificar distintos orígenes
- Opciones:
    - -i => Pregunta antes de sobreescribir
    - -r => Copia directorios recursivamente
    - -u => Copia solamente los ficheros nuevos o modificados
    - -p => Preserva los mismos permisos
    - -a => Es -p, -r y -u juntos
- Ejemplos

RMDIR

- Sistaxis: rmdir
- Borrar directorios SOLO VACÍOS

RM

- Sistaxis: rm [opciones] [ficheros|directorios/patron]
- Borra ficheros y/o directorios
- Opciones
    - -r => Borra recursivamente
    - -i => Pregunta antes de borrar
    - -f => Fuerza eliminación

STAT

- Sintaxis: stat
- Muetra información sobre el fichero

FIND

- Sintaxis: find directorio/s [opciones]
- Busca ficheros o directorios
- Opciones:
    - --name patrón => Busca el patrón en los nombres de ficheros y directorios
    - --iname patrón => Busca el patrón en los nombres de ficheros y directorios sin tener en cuenta mayúsculas o minúsculas
    - --type f|d|l => Filtra la búsqueda por ficheros (f), directorios (d) y links (l)
    - --size +|-tamañoK|M|G => Filtra la búsqueda por tamaño, (+) ficheros de mayores a ese tamañó, (-) ficheros menores a el tamaño, (K) Kilobytes, (M) Megabytes, (G) Gigabyte
    - --atime +|-dias => Filtra la búsqueda por los tiempos de acceso al fichero en días, (+) ficheros con tiempo de acceso mayores al número de días, (-) ficheros con tiempo de acceso menores al número de días
    - --mtime +|-dias => Filtra la búsqueda por los tiempos de modificación al fichero en días, (+) ficheros con tiempo de modificación mayores al número de días, (-) ficheros con tiempo de modificación menores al número de días
    - --ctime +|-dias => Filtra la búsqueda por los tiempos de cambio al fichero en días, (+) ficheros con tiempo de cambio mayores al número de días, (-) ficheros con tiempo de cambio menores al número de días
    - --atime minutos => Lo mismo pero en minutos
    - --mtime minutos => Lo mismo pero en minutos
    - --ctime minutos => Lo mismo pero en minutos
    - --user usuario => Filtra los ficheros, directorios o enlaces por su propietario
    - --group grupo => Filtra ficheros, directorios o enlaces por su grupo
    - --maxdepth numero => Realiza la búsqueda en el número de subdirectorios indicado

- Ejemplos:

```
# Busca los ficheros con extensión ".txt" en el directorio "/tmp"
$ find /tmp --name "*.txt"

# Busca los ficheros que contengan us o US
$ find . --iname "*US*"
```

MV

- Sintaxis: mv [opciones] origen y destino
- Mueve los ficheros de origen a un destino
- Si las carpetas de origen y de destino son la misma renombra
- Opciones:
    - -i => Pregunta antes de sobrescribir
    - -u => Solo mueve los ficheros nuevos o modificados

CHMOD

- Sintaxis: chmod permisos ficheros/directorios
- Permite cambiar permisos de ficheros/directorios
- Opciones:
    - -R => Cambia el propietario y grupo recursivamente
    - -c => Muestra los cambios que realmente hace
    - -v => Muestra todo, los cambios efectuados y los que no necesita realizar
- Los permisos pueden ser octal/modo

Octal	Modo	También
640	u=rw,g=r,o=	660 -> g+w
755	u=rwx,g=rx,o=	750 -> o-rx

- Ejemplos:

```
chmod 644 fichero.txt
chmod -v u=rw,g=rw,o=r fichero.txt
```

CHOWN

- Sintaxis: chown [opciones] propietario:grupo fichero/carpeta [ficheros/carpetas]
- Cambia el propietario y grupo de un fichero o carpeta
- Este comando solomente es posible realizarlo con permisos de administrador
- Opciones:
    - -R => Cambia el propietario y grupo recursivamente
    - -c => Muestra los cambios que realmente hace
    - -v => Muestra todo, los cambios efectuados y los que no necesita realizar

CHGRP

- Sintaxis: chgrp [opciones] grupo fichero/carpeta [ficheros/carpetas]
- Opciones:
    - -R => Cambia el propietario y grupo recursivamente
    - -c => Muestra los cambios que realmente hace
    - -v => Muestra todo, los cambios efectuados y los que no necesita realizar

WHICH

- Sintaxis: which comando
- Muestra la ruta al comando

WHEREIS

- Sintaxis: whereis comando
- Muestra la ruta al comando y la ruta al fichero de ayuda el man

APT

- Sintaxis: apt [subcomando]
- Su nombre viene de Advanced Packing Tool
- Utilizada por Debian y sus derivados
- Instala los programas y librerías en binario
- Facilidad para actualizar la distribución
- Subcomandos
    - update => Actualiza el listado de software disponible en los repositorios
    - upgrade => Actualiza programas y librerías no actulizadas
    - dist-upgrade => Actualiza también librerías del sistema
    - autoremove => Borraría dependencias que no se están utilizando
- Ejemplos:

```
# Actuliza la lista de software disponible en los repositorios
apt update

# Actuliza programas y librerías
apt upgrade

# Actualiza todo
apt dist-upgrade

# Borra las dependencias no utilizadas
apt autoremove
```

APT-CACHE

- Sintaxis: apt-cache search [opciones] expresión
- Utilidad para consultar la base de datos local que contiene la lista de paquetes
- Sin opciones busca en el nombre del paquete, su descripción corta y su descripción larga
- Opciones: --names-only => Busca solo en el nombre del paquete
- Ejemplos:

```
# Busca todos los paquetes que tengan la palabra "
apt-cache search ftp
```

APT-SHOW

- Sintaxis: apt-show paquete
- Muestra información sobre el paquete

DPKG

- Sintaxis: dpkg [opciones] paquete
- En los derivados de REDHAT la utilidad que se utiliza es rpm
- Permite installar paquetes sin sus depencias
- Opciones:
    - -i paquete => Instala
    - l => lista los paquetes instalados

SHUTDOWN

- Sintaxis: shutdown [opciones] [tiempo]
- Apaga o reinicia el equipo
- Si no se indica el tiempo por defecto apagará o reiniciará en un minuto
- Opciones:
    - -r => reinicia el equipo
    - -h => Apaga el equipo
    - -c => Cancela el apagado programado
    - -k => No apaga solo avisa a los usuarios
- Ejemplos:

```
# Apaga el equipo en un minuto
$ shutdown -h

# Reinicia el equipo en un minuto
$ shutdown -r

# Reinicia inmdiatamente
$ shutdown -r now

$ shutdown -r now

# Apaga en 5 minutos
$ shutdown -h +5
```

```
# Apaga a las 14:40
$ shutdown -h 14:40
```

SLEEP

- Sintaxis: sleep segundos
- Lanza un proceso que no hace nada durante el número de segundos indicado
- Ejemplos:

```
$ sleep 30

# Lanza el sleep durante 60 segundos y después el updatedb
$ sleep 60 && updatedb
```

JOBS

- Sintaxis: jobs
- Cuando lanzamos un comando toma el control de la sesión actual. Si el comando que estamos ejecutnado va a tardar uno, dos o más minutos tenemos que terminar a que termine, una posible solución es abrir varias sesiones
- Ejemplo

```
# El mensaje "Terminado" se muestra por pantalla cuando pasa un minuto,
mientras tanto no podemos ejecutar otro comando
$ sleep 60 && echo "Terminado"
```

- Otra solución para ejecutar las aplicaciones en paralelo en una misma consola es ejecutar las aplicaciones en segundo plano mediante el caracter `&`
- Ejemplo:

```
# Lanza el comando sleep en segundo plano
$ sleep 60 &
```

- Cuando lanzamos el comando nos muestra dos identificadores `[1] 1484`, el primero es el identificador en la tabla de procesos del usuario, el segundo identificador es el PID del proceso
- Mediante el comando jobs podemos ver la tabla de procesos

- Ejemplo:

```
# Muestra la tabla de procesos en segundo plano del usuario
$ jobs
```

- Muestra

Id Tabla de procesos	Estado	Comando
[1]	Ejecutando	sleep 60
[2]-	Ejecutando	sleep 120
[3]+	Ejecutando	sleep 180

- Si la lista se vacía se reinica la numeración del identificador de la tabla de procesos en segundo plano
- El símbolo "+" indica que es el último proceso o que cambia su estado en la tabla de procesos
- El símbolo "-" indica que es el penúltimo proceso o que cambia su estado en la tabla de procesos

FG

- Sintaxis: fg [%numero]
- Envía un proceso que se está ejecutando en segundo plano a primer plano
- El `numero` es opcional e indica el identificador del proceso en la tabla de procesos que se quiere mandar a primer plano, si no se indica número se envíara a primer plano el último proceso que se insertó en la tabla de procesos ("+")
- Ejemplos

```
# Manda a primer plano el último proceso que se insertó o cambió de estado en
la tabla de procesos
$ fg

# Manda a primer plano el proceso con identificador en la tabla de procesos
igual a 1
$fg %1
```

BG

- Sintaxis: bg [%n]
- Envía un proceso a segundo plano
- Si quiero poner un proceso que está en primer plano en segundo plano debo realizar dos pasos
    i. `ctrl+z` => Lo pone en estado suspendido (stopped) y lo inserta en la tabla de procesos
    ii. `bg` => Lo ejecuta en segundo plano
- El `numero` es opcional e indica el identificador del proceso en la tabla de procesos que se quiere ejecutar en segundo plano, si no se indica el `numero` se ejecutará en segúndo plano el último proceso que entró o cambió su estado en la tabla de procesos
- Debemos tener cuidado con los comandos en segundo plano que siguen mostrando la salida por consola ya que dificultará la introducción de nuevos comandos
- Ejemplos:

```
# Ejecutadomos sleep en primer plano
$ sleep 60

# Para pausarlo e introducirlo en la tabla de procesos en segundo plano
ctrl+z

# Después relanzamos el proceso
$ bg

# Como acabamos de introducirlo en la tabla de procesos no es necesario el
indicar el identificador del proceso en la tabla de procesos
```

PS

- Sintaxis: ps [opciones]
- Muestra información sobre los procesos activos
- Por defecto sin nunguna opción muestra:
    - PID => Identificador del proceso en el sistema
    - TTY => Terminal asociado si lo tiene
    - TIME => Tiempo de ejecución (Tiempo de CPU consumido)
    - CMD => Comando ejecutado (nombre del proceso)

- Entre las opciones podemos distingur dos formas con "-" y sin él
    - a => Muestra todos los procesos asociados a un terminal (sin guión)
    - x => Muestra todos los procesos del sistema (sin guión)
    - -e => Muestra información parecida a los dos anteriores (con guión)
    - u => Información orientada al usuario
    - -f => Formato lago
- Entre la información en formato largo se encuentra:
    - UID => Usuario que ejecuta el proceso
    - PID => Identificador del proceso padre
    - C => Uso del procesador
    - STIME => Hora de inicio de la ejecución
- Entre la información orientada al usuario se encuentra:
    - USER => Usuario propietario del proceso
    - %CPU => Porcentaje de uso del procesador
    - %MEM => Porcentaje de uso de memoria
    - VSZ => Memoria virtual utilizada por el proceso
    - RSS => Memoria física utilizada por el proceso
    - STAT => Esado del proceso
    - START => Hora de inicio del proceso
- Los posibles estados del proceso son:
    - S => Esperando
    - R => Ejecutando
    - D => Esperando por algún dispositivo
    - T => Pausado
    - Z => No responde
- Los posibles estados del proceso suelen ir acompañados de información adicional:
    - s => Proceso padre
    - l => Proceso con hilos
        -
        - => Primer plano

KILL

- Sintaxis: kill [-señal] PID
- Manda una señal a un proceso mediante su identificador
- Las posibles señales son

Número de señal	Nombre	Descripción
15	SIGTERM	Terminar el proceso
9	SIGKILL	Matar el proceso
17,19,23	SIGSTOP	Pausarlo
3	SIGQUIT	Salir desde teclado ctrl+\
2	SIGINT	Salir desde teclado ctrl+c
1	SIGHUB	Utilizado para recargar configuraciones

- Por defecto si no indicamos señal se manda la señal 15
- Para indicar la señar que queremos mandar podemos indicar su número, o su nombre, o su nombre sin la el prefico SIG
- Ejemplos:

```
# Envía la señal SIGTERM al proceso con identificador 1467
kill 1467

# Mata el proceso 1489
kill -9 1489

# Pausa el proceso 1490
kill -stop 1490
```

KILLALL

- Sintaxis: killall [-u usuario] [-señal] [nombre_proceso]
- Envía una señal de las indicadas en el apartado anterior a todos los procesos con un nombre concreto o de un usuario concreto o ambos
- Este comando se puede ejecutar para matar tus propios procesos, pero si se utiliza la opción `-u usuario` con otro usuario es necesario privilegios de administración

```
# Manda la señal por defecto (SIGTERM) a todos los procesos sleep
killall sleep

# Manda la señal (SIGKILL) a todos los procesos de german
killall -u german -9

# Manda la señal (SIGKILL) a todos los procesos sleep del usuario german
killall -u german -9 sleep
```

FREE

- Sintaxis: free [opciones]
- Muestra información relativa a la memoria física y virtual
- Opciones:
    - -m => Muestra la información en MB
    - -g => Muestra la información en GB
    - -h => Muestra la información en "Human Readable"
- Muestra:
    - Memoria física:
        - Total
        - Usada
        - Libre
        - Shared => Memoria compartida entre procesos
        - Buffers => Tamaño de los buffers de memoria
        - Cache => Tamaño de la memoria cache
    - Swap (area de intercambio / memoria virtual)
        - Total
        - Usada
        - Libre

```
# Muestra la información en un formato más agradable para el usuario
$ free -h
```

UPTIME

- Sintaxis: uptime
- Muestra:
    - Cuanto tiempo lleva el sistema encendido
    - Número de usuarios conectados
    - Carga del sistema
- En cuanto a la carga del sistema muestra la media del último minuto, de los últimos cinco minutos, y la media de los últimos 15 minutos
- Ejemplo:

```
$ uptime
```

TOP

- Sintaxis: top [opciones]
- Muestra en tiempo real información del sistema
- Tiene dos apartados
    - Información del sistema
    - Lista de procesos
- Opciones:
    - -d segundos => Actualiza la información cuando se cumplen el númerod de segundos indicados, por defecto 3 segundos
    - -u usuario => Muestra la lista de procesos de un usuario en concreto
- Información:
    - Primera línea => Comando `uptime`
    - Segunda línea => Número de procesos, número de procesos dormidos, número de procesos ejecutandose, ...
    - Tercera línea:

        - us => % de uso del procesador
        - sy => % de uso de los procesos del sistema
        - id => % de procesos que están esperando a acceder al procesador
        - % de procesos que han cambido su prioridad
    - Cuarta y quinta línea: Comando `free`
    - Siguientes líneas salida del comando `ps` pero ordenadas por el uso del procesador

- Información de la lista de procesos

    - NI => significa prioridad y va desde -19 hasta 20, siendo 20 el valor con menos prioridad, 0 el valor por defecto y -19 el valor de mayor prioridad
    - VIRT => Memoria virtual
    - RES => Memoria estática
    - SMR => Memoria compartida
    - %CPU
    - %MEMORIA
    - TIME+`=> Tiempo que ha utilizado el procesador
    - COMANDO

- Opciones interactivas: -h => Nos muesta la ayuda -z => Nos muestra colores -k => Sirva para matar un proceso -M => Ordena los proceso por su memoria consumida

NICE

- Sintaxis: nice -n prioridad comando
- Ejecuta un comando con una prioridad distinta a la de por defecto
- La prioridad por defecto es 0, la máxima prioridad es -19 y la menor prioridad es 20
- Ejemplos:

```
# Error ya que es una prioridad por debajo de la de por defecto, necesitamos privilegios de administrador
$ nice -n -5 sleep 1800 &

# Lanzamos el sleep con prioridad 5
$ nice -n 5 sleep 1800 &
```

RENICE

- Sintaxis: renice -n prioridad PID
- Para cambiar la prioridad de un proceso que ya se esté ejecutando
- En ambos comandos `nice` y `renice` solo podemos disminuir la prioridad (aumentar el número), para aumentar la prioridad es necesario privilegios de administrador
- Ejemplos:

```
# Establecemos la prioridad a un proceso con PID 1349 a 10, debe ser mayor
que la que ya tenía
$ renice -n 10 1349
```