

# Sistemas Operativos

## Ejercicios de Unix

18/02/2026

## Contenidos

<b>1. Primeros pasos en unix</b>	<b>3</b>
1.4 Fichero de <b>man</b>	3
1.10 Última orden del historial	3
1.11 Órdenes con “p”	4
1.12 Última orden en bash como shell	4
1.13 Órdenes internas y externas	4
1.14 Orden <b>what is</b>	5
1.15 <b>man</b> como <b>apropos</b>	5
1.16 Más partido a <b>history</b>	5
<b>2. El sistema de ficheros</b>	<b>7</b>
2.1 Volver al directorio de trabajo	7
2.2 Rutas relativas y absolutas	7
2.3 Contenido del directorio raíz	7
2.4 Enlace a un archivo	8
2.7 Permisos en directorio	8
2.8 Directorio de solo acceso	9
2.9 Expresión regular con “.”	9
2.10 Inspección de ciertos ficheros	10
2.11 Enlaces en HOME	10
2.14 Archivos cuyo nombre termina con dígito	11
2.15 Archivos que no acaben en “.c”	11
2.16 Ficheros de 2 o más caracteres de nombre	11
2.17 Ficheros . y	11
2.18 Directorio <b>/dev</b>	12

2.19	Orden <code>df</code>	12
2.20	Orden <code>mount</code>	13
2.21	Permisos mínimos para crear un archivo en un directorio	13
2.22	Permisos mínimos de ejecución	14
2.23	Permisos para eliminar un fichero	14
2.24	Permisos para leer un fichero	14
2.25	Permisos para modificar un fichero	15
2.26	Borrar todos los ficheros del directorio actual	15
2.27	Borrar un subdirectorio y su contenido	16
2.28	Cambiar directorio actual a uno de sus hermanos	16
2.29	Conocer información del directorio actual	17
2.30	Información del directorio padre al actual	17
2.31	Copiar todos los ficheros del actual a otro directorio	17
2.32	Crear hijo de otro directorio	18
2.33	Mover ficheros del directorio actual a otro	18
<b>3.</b>	<b>Procesos y usuarios</b>	<b>19</b>
3.1	Usuarios conectados actualmente	19
3.2	Diferencia entre órdenes con <code>sleep</code>	19
3.3	Terminación de programas en primer y en segundo plano	19
3.6	Interpreta los campos de la orden <code>top</code>	19
3.7	Suspender y reanudar un proceso	20
3.8	Conocer procesos que el usuario tiene en ejecución	21
3.9	Lanzar procesos en segundo plano	22
3.10	Reanudar en segundo plano un proceso suspendido	22
3.11	Suspender el proceso en primer plano	22
3.12	Lista de procesos que hemos lanzado en segundo plano	22
3.13	Suspender un proceso de otra terminal	23
3.14	Alternativas para eliminar procesos en ejecución	23
<b>4.</b>	<b>Conductos, Filtros y Redirecciones de E/S</b>	<b>24</b>
4.1	Eliminar mensajes de otros usuarios	24
4.2	Crear un fichero usando <code>cat</code>	24
4.3	Visualizar el fichero de contraseñas	24

## §1. Primeros pasos en unix

### 1.4 Fichero de man

¿Qué fichero se ejecuta cuando introducimos la orden man?

Usando type:

```
U0299855@Ubuntu-24:~/EjerciciosShell$ type man
man is /usr/bin/man
```

Veamos qué pasa al ejecutar este archivo directamente:

```
U0299855@Ubuntu-24:~/EjerciciosShell$ /usr/bin/man
What manual page do you want?
For example, try 'man man'.

U0299855@Ubuntu-24:~/EjerciciosShell$ /usr/bin/man --
help
Usage: man [OPTION...] [SECTION] PAGE...

-C, --config-file=FILE      use this user configuration
                             file
-d, --debug                  emit debugging messages
...
```

En efecto, se corresponde con el comportamiento esperado al usar directamente el comando man.

### 1.10 Última orden del historial

La orden history, ¿es interna o externa?

Comprobamos con type:

```
U0299855@Ubuntu-24:~/EjerciciosShell$ type history
history is a shell builtin
```

#### Reproduce la última orden introducida

Tras consultar la ayuda de la orden con `help history`, sabemos que es posible utilizar índices negativos para acceder a entradas del historial. En tal caso, comenzará a contarse por el final de la lista:

```
U0299855@Ubuntu-24:~/EjerciciosShell$ echo Orden mas
reciente
```

```
Orden mas reciente
U0299855@Ubuntu-24:~/EjerciciosShell$ !-1
echo Orden mas reciente
Orden mas reciente
U0299855@Ubuntu-24:~/EjerciciosShell$
```

## 1.11 Órdenes con “p”

Utilizando las facilidades del shell, averigua qué órdenes externas comienzan con la letra p. ¿Cuántas por “pa”? ¿Y por “pas”?

Pulsando dos veces la tecla del tabulador para completar:

```
U0299855@Ubuntu-24:~/EjerciciosShell$ p
Display all 148 possibilities? (y or n)
U0299855@Ubuntu-24:~/EjerciciosShell$ pa
pager                pam_extrausers_update
    pam_timestamp_check    partx
    pastebinit
pam-auth-update      pam_getenv                parted
                        passwd                    patch
pam_extrausers_chkpwd  pam_namespace_helper    partprobe
                        paste                        pathchk
U0299855@Ubuntu-24:~/EjerciciosShell$ pas
passwd    paste    pastebinit
U0299855@Ubuntu-24:~/EjerciciosShell$
```

## 1.12 Última orden en bash como shell

¿Cómo puedo repetir la ultima orden que he introducido, si estoy utilizando el bash como shell?

Si estamos en bash interactivo, podemos usar `!-1` ó `!!` para ejecutar la última orden, como en el ejercicio 1.10. Esto no funciona en bash de scripts.

## 1.13 Órdenes internas y externas

¿Con qué orden puedo saber si una orden es interna o externa del shell?  
¿Cómo se utiliza?

Usando el comando `type`, podemos saber si una orden es interna (*builtin*) o externa. Podemos obtener información sobre las primeras con `help` y de las segundas con `man`.

1. Ejemplo de orden interna
2. Ejemplo de orden externa

## 1.14 Orden `whatis`

La orden `whatis` nos da información muy resumida sobre la función de otras órdenes, ya sean internas o externas:

```
U0299855@Ubuntu-24:~$ whatis history
history (3readline) - GNU History Library

U0299855@Ubuntu-24:~$ whatis man
man (1) - interfaz de los manuales de referencia del
sistema
```

Se nutre de entradas en el `man` (en concreto, devuelve el apartado `NAME`); luego es necesario que la orden a consultar tenga una entrada en él.

## 1.15 `man` como `apropos`

¿Qué opción del `man` hace que funcione de manera similar a la orden `apropos`?

Podemos usar la opción `-k` para que funcione de manera equivalente a `apropos`. Además, la opción `-K` cumple esta función, pero además buscando en todas las páginas (`apropos` solo busca en `NAME`, de donde también se nutre `whatis`).

## 1.16 Más partido a `history`

Ejecutar comando de hace 6 órdenes y la última que empiece por “hi”

En primer lugar, podemos usar `!-6`

```
U0299855@Ubuntu-24:~$ history 10
613  echo 2
614  echo 3
```

```
615 echo 4
616 echo 5
617 echo 6
618 echo 7
619 echo 8
620 echo 9
621 echo 10
622 history 10
U0299855@Ubuntu-24:~$ !-6
echo 6
6
```

Para ejecutar la orden más reciente que comience por “hi”, usamos !hi

```
U0299855@Ubuntu-24:~$ !hi
history 10
617 echo 6
618 echo 7
619 echo 8
620 echo 9
621 echo 10
622 history 10
623 echo 6
624 clear
625 man history
626 history 10
```

Si en su lugar queremos ejecutar la orden más reciente que contenga el substring “hi”, podemos usar !?hi:

```
U0299855@Ubuntu-24:~$ man history
U0299855@Ubuntu-24:~$ !?hi
man history
NAME
    history - GNU History Library
...
```

## §2. El sistema de ficheros

### 2.1 Volver al directorio de trabajo

Vete al directorio raíz y utiliza la orden `cd` de la manera más sencilla posible para volver a tu directorio de trabajo

```
U0299855@Ubuntu-24:~$ cd /
U0299855@Ubuntu-24:/$ cd
U0299855@Ubuntu-24:~$
```

### 2.2 Rutas relativas y absolutas

Accede a tu directorio de trabajo desde `/etc` con rutas relativas y absolutas

Usando las dos formas:

```
U0299855@Ubuntu-24:~$ ls
archivo1.txt      archivo.txt      claves.txt
U0299855@Ubuntu-24:~$ pwd
/home/U0299855
U0299855@Ubuntu-24:~$ cd /etc
U0299855@Ubuntu-24:/etc$ ls /home/U0299855
archivo1.txt      archivo.txt      claves.txt
U0299855@Ubuntu-24:/etc$ ls ../home/U0299855
archivo1.txt      archivo.txt      claves.txt
```

### 2.3 Contenido del directorio raíz

Sitúate en el directorio raíz. Comprueba qué contenido tiene. ¿Existen directorios?

Contenido total:

```
U0299855@Ubuntu-24:~$ ls -l /
total 84
drwxr-xr-x  2 root root  4096 feb 26  2024 bin.usr-is-merged
drwxr-xr-x  5 root root  4096 feb 10 06:41 boot
drwxr-xr-x 18 root root 4040 feb 13 13:47 dev
...
```

Podemos saber que hay directorios al haber entradas que comienzan por `d`.

## 2.4 Enlace a un archivo

Usando `ln` sin la opción `-s` crearemos un enlace directo (*hard*) a un archivo.

```
U0299855@Ubuntu-24:~/EjerciciosShell$ echo Hola > f1.txt
U0299855@Ubuntu-24:~/EjerciciosShell$ ln f1.txt f2.txt
U0299855@Ubuntu-24:~/EjerciciosShell$ ls -l
total 8
-rw-rw-r-- 2 U0299855 U0299855  5 feb 19 10:24 f1.txt
-rw-rw-r-- 2 U0299855 U0299855  5 feb 19 10:24 f2.txt
U0299855@Ubuntu-24:~/EjerciciosShell$ echo caracola >>
    f1.txt
U0299855@Ubuntu-24:~/EjerciciosShell$ cat f2.txt
Hola
caracola
U0299855@Ubuntu-24:~/EjerciciosShell$
```

Nótese el número 2 en el listado de los archivos, que indica que ambos referencian la misma información.

## 2.7 Permisos en directorio

Crear un directorio de prueba. Copia algunos ficheros a su interior. Modifica sus permisos de éste para que sólo el propietario pueda leerlo. Intenta situarte en él. ¿Puedes? ¿Por qué?. Intenta ver su contenido. ¿Puedes? ¿Por qué?

```
$ mkdir prueba
$ cp f* prueba/
$ chmod 400 prueba
$ cd prueba
bash: cd: prueba: Permission denied
$ ls prueba
ls: cannot access 'prueba/f1.txt': Permission denied
ls: cannot access 'prueba/f2.txt': Permission denied
f1.txt  f2.txt
$ dir prueba
f1.txt  f2.txt
```

No es posible entrar directamente al directorio, pues esta operación requiere permisos de ejecución (x), que no tenemos. Sí podemos ver los nombres de archivos al requerir esto solo lectura.



La orden `ls` nos da error, ya que intenta mostrar estadísticas de los archivos, que requiere de nuevo permisos de ejecución. El comando `dir` solo necesita permisos de lectura, luego funciona.

## 2.8 Directorio de solo acceso

Cambia los permisos del directorio anterior para que sólo el propietario pueda acceder a él. Intenta situarte en él. ¿Puedes? ¿Por qué?. Intenta ver su contenido. ¿Puedes? ¿Por qué?

Continuando con el ejercicio anterior,

```
$ chmod 100 prueba
$ cd prueba
$ dir
dir: cannot open directory '.': Permission denied
$ cat f1.txt
Hola
caracola
```

En este caso, ocurre justamente lo contrario: Podemos situarnos en el directorio, pero no listar sus contenidos. Sí podemos acceder a los archivos si sabemos su nombre, tal y como hemos hecho con `cat`.

Expresión regular de tres caracteres **Indica la expresión regular que hay que introducir para referirnos a todos los ficheros cuyo nombre contenga tres caracteres, siendo el primero necesariamente una letra (mayúscula o minúscula).**

Podemos usar la siguiente expresión: `[a-zA-Z]??`

```
$ ls
3ff  f3789  f3A  fAB
$ ls [a-zA-Z]??
f3A  fAB
```

## 2.9 Expresión regular con “.”

Indica la expresión regular que hay que introducir para referirnos a todos los ficheros cuyo nombre contenga un “.” en cualquier posición a excepción de la primera.

Podemos usar la expresión `[!..]*.*`

```
$ ls -a
. .. buenos.dias bDias buenos.dias.hello
.secret .cia.txt
$ ls -a [!..]*.*
buenos.dias buenos.dias.hello
```

## 2.10 Inspección de ciertos ficheros

Comprobando los contenidos para mi usuario:

```
$ grep 299855 /etc/passwd
U0299855:x:1166:1166::/home/U0299855:/bin/bash
$ grep 1166 /etc/group
U0299855:x:1166:
```

En este caso, cada usuario tiene un grupo de nombre igual al de su nombre de usuario, y los IDs de usuario y grupo también coinciden. podemos comprobarlo con `id`:

```
$ id
uid=1166(U0299855) gid=1166(U0299855) groups=1166(U0299855)
```

## 2.11 Enlaces en HOME

**Ve a tu directorio HOME. Mira cuántos enlaces tiene. Crea un directorio dentro de él. Comprueba cuántos enlaces tiene ahora tu directorio HOME. ¿Por qué ha cambiado así?**

```
$ ls -ld ~
drwx----- 9 U0299855 U0299855 4096 feb 19 11:14 /home/
U0299855
$ mkdir ~/OtroMas
$ ls -ld ~
drwx----- 10 U0299855 U0299855 4096 feb 19 11:15 /home/
U0299855
```

Al crear el directorio, el número ha subido a otro más. Tal y como hemos podido ejecutar en el [ejercicio 2.9](#), al hacer `ls -al` podemos comprobar que cada directorio tiene los enlaces `.` (directorio actual) y `..` (directorio padre). Al crear una

subcarpeta en HOME, el enlace .. de OtroMas apunta a HOME, incrementando por tanto en uno el número de enlaces que apuntan a ella.

## 2.14 Archivos cuyo nombre termina con dígito

Indica la expresión regular que hay que introducir para referirnos a todos los ficheros cuyo nombre comience por "a" y terminen por un dígito.

Podemos usar la expresión: `a*[1-9]`

```
$ dir
2a2  a1  a2  a2d  adosa2
$ ls a*[1-9]
a1  a2  adosa2
```

## 2.15 Archivos que no acaben en ".c"

Expresión: `!(*.c)`

```
$ dir
c  c.c  holac  hola.c.txt
$ ls !(*.c)
c  holac  hola.c.txt
```

## 2.16 Ficheros de 2 o más caracteres de nombre

Expresión: `??*`

```
$ dir
c  c.c  holac  hola.c.txt
$ ls ??*
c.c  holac  hola.c.txt
```

## 2.17 Ficheros . y ..

Inspeccionando los ficheros que tengo en mi directorio me he encontrado con que tengo uno que se llama "." y otro "..". Yo no los he

creado y no sé lo que significan. ¿Quién los ha creado, y qué utilidad tienen?

Son los enlaces al directorio actual y al directorio padre. Se crean de manera automática, tal y como hemos visto en el ejercicio 2.11.

## 2.18 Directorio /dev

Ve al directorio /dev y comprueba que contiene ficheros de tipo "dispositivo". Comprobar también que se da información acerca del número mayor y menor justo antes de la fecha de modificación.

Al hacer `ls -l`, podemos ver que algunas entradas comienzan por "c".

```
$ ls -l | grep ^c
crw-r--r--  1 root root      10, 235 feb 13 13:47 autofs
crw-rw----  1 root disk     10, 234 feb 13 13:47 btrfs-
  control
crw--w----  1 root tty       5,   1 feb 13 13:48 console
...
```

Estos son dispositivos de caracteres, aunque existen otros tipos. El número mayor indica el controlador o tipo de dispositivo, mientras que el otro número indica el dispositivo en particular que está siendo gestionado por ese controlador.

## 2.19 Orden df

La orden `df`, de *disk free* indica el tamaño disponible en disco:

```
$ df
Filesystem            1K-blocks    Used Available Use% Mounted
on
tmpfs                  1634560      1568    1632992   1% /run
/dev/xvda1             29378688 26430712    2931592  91% /
tmpfs                  8172784        0    8172784   0% /dev/
shm
tmpfs                   5120         0        5120   0% /run/
lock
/dev/xvda16            901520     119220    719172   15% /boot
/dev/xvda15            106832      6250    100582    6% /boot/
efi
tmpfs                  1634556       16    1634540   1% /run/
user/1006
```

```
tmpfs          1634556      16    1634540    1% /run/
user/1003
...
```

## 2.20 Orden mount

Esta orden permite montar un disco de hardware adicional para su uso en el SO. No es práctica para usar por el usuario. Si se usa sin parámetros, dirá qué discos están montados y dónde.

## 2.21 Permisos mínimos para crear un archivo en un directorio

Necesitamos tener permisos de ejecución y de escritura en el directorio:

```
$ mkdir prueba
$ chmod 000 prueba
$ echo Hola > prueba/test.txt
bash: prueba/test.txt: Permission denied
$ chmod 200 prueba
$ echo Hola > prueba/test.txt
bash: prueba/test.txt: Permission denied
$ chmod 300 prueba
$ echo Hola > prueba/test.txt
$
```

Veamos qué ocurre con los permisos de fichero:

```
$ umask 0777
$ cd ..
$ chmod 300 prueba
$ echo hola > holaTest.txt
$ echo hola > holaTest.txt
bash: holaTest.txt: Permission denied
$
```

No hace falta tener permiso ninguno para escribir, pero sí para remplazar.

En conclusión, para crear un archivo en un directorio es necesario tener **w** y **x** en el directorio, pero no se necesita permiso ninguno para el fichero. Aun creando un fichero dentro del directorio actual, es necesario tener **x** para ello.

## 2.22 Permisos mínimos de ejecución

Necesitamos *r*, *x* en los permisos de fichero, y *x* en los de directorio:

```
$ chmod 100 prueba
$ cd prueba
$ ./ejecutable.bash
Hola
... # Cambio de permisos para ls
-r-x----- 1 U0299855 U0299855 10 feb 19 12:04
    ejecutable.bash
```

Si quitásemos alguno de estos permisos, la ejecución no podría llevarse a cabo.

## 2.23 Permisos para eliminar un fichero

En cuanto a ficheros, no es necesario tener ningún permiso, aunque `rm` nos dará un prompt en caso de que no podamos escribir en ellos:

```
$ chmod 000 hola.txt
$ rm hola.txt
rm: remove write-protected regular file 'hola.txt'? y
$ dir
$
```

En cuanto a permisos de carpetas, son necesarios *write* y *execute*:

```
$ echo Hola > hola.txt
$ chmod 200 .
$ rm hola.txt
rm: cannot remove 'hola.txt': Permission denied
...
$ chmod 100 .
$ rm hola.txt
rm: cannot remove 'hola.txt': Permission denied
...
$ chmod 300 .
$ rm hola.txt
$
```

## 2.24 Permisos para leer un fichero

Permisos de fichero: basta con *read*.

```
$ chmod 000 hola.txt
$ cat hola.txt
cat: hola.txt: Permission denied
$ chmod 400 hola.txt
$ cat hola.txt
Hola
$
```

Permisos de directorio: basta con **x**, y **r** no sirve.

```
$ chmod 400 .
$ cat hola.txt
cat: hola.txt: Permission denied
...
$ chmod 100 .
$ cat hola.txt
Hola
$
```

## 2.25 Permisos para modificar un fichero

Basta tener permiso de ejecución a nivel de carpeta, y permiso de escritura a nivel de archivo:

```
$ chmod 100 .
$ chmod 200 hola.txt
$ echo buenos dias >> hola.txt
...
$ cat hola.txt
Hola
buenos dias
$
```

## 2.26 Borrar todos los ficheros del directorio actual

Suponiendo que tengamos los permisos necesarios, basta usar **rm \***

```
$ dir
1.txt  2.txt  3.txt  hola.txt
$ rm *
```

```
$ dir
$
```

## 2.27 Borrar un subdirectorio y su contenido

Quiero borrar un directorio "d1", perteneciente al directorio actual, y todo su contenido. Suponiendo que tengo los permisos necesarios, ¿cómo puedo hacerlo mediante una única orden?

```
$ ls -l
total 4
-rw-r--r-- 1 javier javier    0 Feb 19 19:45 a.out
drwxr-xr-x 3 javier javier 4096 Feb 19 19:47 d1
$ ls -l ./d1
total 16
-rw-r--r-- 1 javier javier   11 Feb 19 19:47 ejecutable
-rw-r--r-- 1 javier javier    5 Feb 19 19:47 file2.txt
-rw-r--r-- 1 javier javier    5 Feb 19 19:47 hola.txt
drwxr-xr-x 2 javier javier 4096 Feb 19 19:48 otroDirDentro
$ rm -r d1/*
$ ls
a.out  d1
$ cd d1
$ ls
$
```

Esta opción no borra el propio directorio y puede saltarse archivos "ocultos" como `.archivo`. Si queremos que esto ocurra, hay que usar `rm -rf` (recursivo y forzoso).

## 2.28 Cambiar directorio actual a uno de sus hermanos

Quiero cambiar el directorio actual a un "hermano" (hijo del mismo padre) del actual. El directorio al que quiero cambiar se llama "d1". ¿Cómo se hace con una única orden

Podemos usar el acceso al directorio madre mediante `..`.

```
$ pwd
/home/javier/EjerciciosShell/prueba/actual
$ cd ../d1
$ pwd
```



```
/home/javier/EjerciciosShell/prueba/d1
$
```

## 2.29 Conocer información del directorio actual

Podemos usar la orden `ls -lh .` para ello:

```
$ ls -ldsh .
4.0K drwxr-xr-x 3 javier javier 4.0K Feb 19 20:02 .
$
```

La opción `-sh` muestra el tamaño de manera más legible para un humano. La opción `-d` es para mostrar el propio directorio y no su contenido.

## 2.30 Información del directorio padre al actual

Análogo al ejercicio anterior, pero usando `..`.

```
$ ls -ldsh ..
4.0K drwx----- 5 javier javier 4.0K Feb 19 19:53 ..
$
```

## 2.31 Copiar todos los ficheros del actual a otro directorio

Basta usar `cp * <ruta>` donde `<ruta>` apunta al otro directorio.

```
$ ls -l
total 12
drwxr-xr-x 2 javier javier 4096 Feb 19 20:02 bobo
-rw-r--r-- 1 javier javier   12 Feb 19 20:08 dias.txt
-rw-r--r-- 1 javier javier   16 Feb 19 19:55 pocaCosa.
txt
$ ls -l ../otroDir
total 8
-rw-r--r-- 1 javier javier 12 Feb 19 20:10 dias.txt
-rw-r--r-- 1 javier javier 16 Feb 19 20:10 pocaCosa.txt
$
```

Como no hemos usado la opción `-r`, se omite `bobo` al ser este un directorio.

## 2.32 Crear hijo de otro directorio

Quiero crear un directorio hijo del directorio "d1", que es hijo del directorio actual. Indique cómo se hace con una única orden?

Usamos `mkdir` con una ruta apropiada:

```
$ mkdir d1
$ mkdir d1/OtroHijo
$ dir
d1
$ cd d1
$ dir
OtroHijo
$
```

## 2.33 Mover ficheros del directorio actual a otro

Usamos la orden `mv`:

```
$ ls -l
total 12
drwxr-xr-x 2 javier javier 4096 Feb 19 20:13 carpeta
-rw-r--r-- 1 javier javier  5 Feb 19 20:13 hola.txt
-rw-r--r-- 1 javier javier  5 Feb 19 20:13 otroFichero
.txt
$ mv * ../dir2
$ ls -l
total 0
$ ls -l ../dir2
total 12
drwxr-xr-x 2 javier javier 4096 Feb 19 20:13 carpeta
-rw-r--r-- 1 javier javier  5 Feb 19 20:13 hola.txt
-rw-r--r-- 1 javier javier  5 Feb 19 20:13 otroFichero
.txt
$
```

## §3. Procesos y usuarios

### 3.1 Usuarios conectados actualmente

Basta usar `who`. Al estar trabajando con WSL en mi máquina personal, salgo únicamente yo:

```
$ who
javier pts/1 2026-02-19 19:33
$
```

### 3.2 Diferencia entre órdenes con `sleep`

Ejecuta las siguientes líneas:

- i) `sleep 15; echo He terminado`
- ii) `(sleep 15; echo He terminado)`

La primera opción trabaja en primer plano, dejando un retardo de 15s para después imprimir “He terminado” por pantalla. La segunda opción hace lo mismo pero en segundo plano, dejando al usuario la posibilidad de ejecutar otras órdenes en la terminal mientras espera el otro proceso por el retardo.

### 3.3 Terminación de programas en primer y en segundo plano

Ejecuta el programa `/home/profesores/albizu/publico/bucle` en primer plano. Haz que termine. Haz lo mismo, pero lanzándolo en en segundo plano. TODO, mirar máquina remota

### 3.6 Interpreta los campos de la orden `top`

```
$ top
top - 21:01:00 up 1:28, 1 user, load average: 0.07, 0.02, 0.00
Tasks: 22 total, 1 running, 21 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 99.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7896.2 total, 7407.5 free, 464.3 used, 175.0 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used. 7431.9 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1083	javier	20	0	9316	5504	3328	R	0.3	0.1	0:00.01	top
1	root	20	0	21932	12772	9444	S	0.0	0.2	0:01.75	systemd
8	root	20	0	3136	2024	1920	S	0.0	0.0	0:00.00	init
...											

Veamos qué es cada campo de la tabla:

- PID: identificador de proceso
- USER: usuario al que pertenece el proceso
- VIRT: tamaño de la memoria virtual del proceso en KiB
- RES: tamaño de la memoria física ocupada en KiB
- SHR: tamaño de memoria compartida en KiB
- S: estado del proceso (en ejecución, zombie...)
- %CPU: uso de CPU
- %MEM: uso de memoria (según el campo RES)
- TIME+: tiempo de CPU
- COMMAND: comando asociado al proceso

El párrafo anterior anterior a la tabla resume la información de esta y muestra la carga promedio.

### 3.7 Suspender y reanudar un proceso

**Define una orden para suspender el proceso en primer plano. Ejecuta la orden man y suspende su ejecución. Usa ps para ver el estado en que está. Reanuda la ejecución del proceso.**

Podemos usar **CNTRL+Z** envía la señal 20 al proceso en primer plano (al igual que podría hacerlo **kill**), que actúa como la 19, pero puede ser ignorada por el proceso en ciertas circunstancias:

```

$ man man
man: can't resolve man7/groff_man.7

[1]+  Stopped                  man man
$ fg
man man
$

```

El Stopped se corresponde con lo ocurrido al hacer CNTRL+Z.

### 3.8 Conocer procesos que el usuario tiene en ejecución

Podemos usar la opción `-u` con las órdenes `ps` o `top` para filtrar por usuarios:

```

$ top -u javier
...
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM
   TIME+ COMMAND
  351 javier      20   0    6180    4992    3456 S   0.0   0.1
    0:00.24 bash
  397 javier      20   0   20116   10880    9088 S   0.0   0.1
    0:00.18 systemd
  398 javier      20   0   21156    3520    1792 S   0.0   0.0
    0:00.00 (sd-pam)
  422 javier      20   0    6056    4992    3456 S   0.0   0.1
    0:00.04 bash
  695 javier      20   0    9280    5120    3072 R   0.0   0.1
    0:00.01 top

$ ps -u javier
  PID TTY          TIME CMD
  351 pts/0      00:00:00 bash
  397 ?          00:00:00 systemd
  398 ?          00:00:00 (sd-pam)
  422 pts/1      00:00:00 bash
  696 pts/0      00:00:00 ps
$

```

### 3.9 Lanzar procesos en segundo plano

Basta usar `&` como en el ejercicio 3.1.

### 3.10 Reanudar en segundo plano un proceso suspendido

Podemos usar la orden `bg` para ello.

```
$ (sleep 60; echo He terminado)
^Z
[1]+  Stopped                  ( sleep 60; echo He
    terminado )
$ bg
[1]+ ( sleep 60; echo He terminado ) &
$ echo Otra cosa mientras
Otra cosa mientras
$ echo Esperamos un poco mas...
Esperamos un poco mas...
$ He terminado
```

### 3.11 Suspender el proceso en primer plano

Basta usar `CNTRL+Z`.

### 3.12 Lista de procesos que hemos lanzado en segundo plano

Podemos usar `jobs`:

```
$ (sleep 60; echo Segundo plano 1)&
[1] 810
$ (sleep 60; echo Segundo plano 2)&
[2] 812
$ (sleep 60; echo Segundo plano 3)&
[3] 814
$ jobs
[1]   Running ( sleep 60; echo Segundo plano 1 ) &
[2]-  Running ( sleep 60; echo Segundo plano 2 ) &
[3]+  Running ( sleep 60; echo Segundo plano 3 ) &
$
```

### 3.13 Suspende un proceso de otra terminal

Indica cómo podemos suspender un proceso que no está en primer plano, pero que se ha lanzado desde otra terminal.

Al estar en otra terminal, debemos usar la orden `kill`.

```
javier@MSI:/mnt/c/Users/Javier$ (sleep 600; echo Otra
terminal)
```

Comprobamos de qué proceso se trata usando `top`

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM
		TIME+	COMMAND						
977	javier	20	0	3128	1664	1664	S	0.0	0.0
		0:00.00	sleep						

En este caso será el 977 al ser de `sleep`. Enviaremos ahora una señal:

```
$ kill -19 977
```

Comprobando con el `top` de nuevo,

977	javier	20	0	3128	1664	1664	T	0.0	0.0
		0:00.00	sleep						

### 3.14 Alternativas para eliminar procesos en ejecución

Podemos usar `kill` para enviar una señal de terminación a un proceso del que conocemos su PID. Según qué señal enviemos, el proceso terminará de manera ordinaria o forzosa:

- **SIGTERM (15):** Terminación “ordenada”, permite la liberación de recursos. Es la opción por defecto
- **SIGKILL (9):** Terminación forzosa, no puede ser capturada ni ignorada

Alternativamente, órdenes como `top` nos permiten terminar procesos desde su interfaz introduciendo el PID correspondiente. No obstante, estas utilidades llaman por debajo a `kill`, luego tienen el comportamiento anteriormente descrito.

## §4. Conductos, Filtros y Redirecciones de E/S

### 4.1 Eliminar mensajes de otros usuarios

Basta usar `mesg n`.

### 4.2 Crear un fichero usando cat

Redirigiendo la salida de esta orden,

```
$ cat . 2> ficheroCat.txt
$ cat ficheroCat.txt
cat: .: Is a directory
$
```

### 4.3 Visualizar el fichero de contraseñas

Visualiza a través de una tubería el contenido del fichero `/etc/passwd`, haciendo que la pantalla se detenga cuando ésta se llene.

Una posible opción es usar `cat /etc/passwd | less`, que da lugar a una visualización del contenido con un formato igual al de `man`.

### 4.4 Añade texto a un fichero ya existente

Redirigimos la salida de `echo` así:

```
$ cat ficheroCat.txt
cat: .: Is a directory
$ echo Otra linea mas >> ficheroCat.txt
$ cat ficheroCat.txt
cat: .: Is a directory
Otra linea mas
$
```



## 4.5 Redirecciona una salida de error a un fichero

La redirección del ejercicio 4.1 es la de la salida de error.

## 4.6 Contar usuarios conectados

La orden `who` muestra los usuarios conectados, tomando la información de cada uno una línea. Así, contar el número de líneas equivale a contar el número de usuarios (no hay líneas extra). Para ello, usamos `wc -l` (*word count* por líneas).

```
$ who | wc -l
1
$
```