

## SESION 2

**Ejercicio 2.1. Cree el siguiente árbol de directorios a partir de un directorio de su cuenta de usuario. Indique también cómo sería posible crear toda esa estructura de directorios mediante una única orden (mire las opciones de la orden de creación de directorios mediante man mkdir). Posteriormente realice las siguientes acciones:**

mkdir -p ejercicio1/ejer2 ejercicio1/ejer3 ejercicio1/ejer1/ejer21

a) En Ejer1 cree los archivos arch100.txt, filetags.txt, practFS.ext y robet201.me.

- \$ touch arch100.txt  
- \$ touch filetags.txt  
- \$ touch practFS.ext  
- \$ touch robet201.me

b) En Ejer21 cree los archivos robet202.me, ejer11sol.txt y blue.me.

-\$ touch robet202.me  
-\$ touch ejer11sol.txt  
-\$ touch blue.me

c) En Ejer2 cree los archivos ejer2arch.txt, ejer2filetags.txt y readme2.pdf.

- \$ touch ejer2arch.txt  
- \$ touch ejer2filetags.txt  
- \$ touch readme2.pdf

d) En Ejer3 cree los archivos ejer3arch.txt, ejer3filetags.txt y readme3.pdf.

\$ touch ejer3arch.txt  
\$ touch ejer3filetags.txt  
\$ touch readme3.pdf

e) ¿Podrían realizarse las acciones anteriores empleando una única orden? Indique cómo.

Se haría con la orden touch seguido de los nombres de los archivos a crear junto con su extensión separado por espacios. Por ejemplo:

- \$ touch robet22.me ejer11sol.txt blue.me

**Ejercicio 2.2. Situados en el directorio ejercicio1 creado anteriormente, realice las siguientes acciones:**

Para hacer este ejercicio nos situamos siempre en el directorio ejercicio1

a) Mueva el directorio Ejer21 al directorio Ejer2.

- \$ mv Ejercicio1/Ejer21 Ejercicio2

b) Copie los archivos de Ejer1 cuya extensión tenga una x al directorio Ejer3.

- \$ cp Ejercicio1/\*.\*x\* Ejercicio3

c) Si estamos situado en el directorio Ejercicio2 y ejecutamos la orden ls -la ../\*arch\*, ¿qué archivo/s, en su caso, debería mostrar?

- arch100.txt  
- ejer3arch.txt

# CURSOS DE INGLÉS EN EL EXTRANJERO

TU FUTURO NO TENDRÁ LÍMITES

- ⑤ 41 escuelas de inglés acreditadas: Reino Unido, Irlanda, Estados Unidos, Canadá, Australia y N. Zelanda
- ⑤ 80 años de experiencia en educación internacional
- ⑤ Cursos para todos los objetivos: inglés general, de negocios, preparación de exámenes, larga duración, inglés + trabajo, entre otros



DESCARGA  
EL CATÁLOGO  
GRATUITO

KAPLANINTERNATIONAL.COM/ES

**KAPLAN** INTERNATIONAL  
ENGLISH

# CURSOS DE INGLÉS EN EL EXTRANJERO

TU FUTURO NO TENDRÁ LÍMITES

- ⦿ 41 escuelas de inglés acreditadas: Reino Unido, Irlanda, Estados Unidos, Canadá, Australia y N. Zelanda
- ⦿ 80 años de experiencia en educación internacional
- ⦿ Cursos para todos los objetivos: inglés general, de negocios, preparación de exámenes, larga duración, inglés + trabajo, entre otros



DESCARGA  
EL CATÁLOGO  
GRATUITO

KAPLANINTERNATIONAL.COM/ES

**KAPLAN** INTERNATIONAL  
ENGLISH

Tu nueva cuenta móvil, pensada para ti, en 8 minutos y listo.

**Ejercicio 2.3. Si estamos situados en el directorio Ejer2, indique la orden necesaria para listar sólo los nombres de todos los archivos del directorio padre.**

\$ ls ../\*.\*

**Ejercicio 2.4. Liste los archivos que estén en su directorio actual y fíjese en alguno que no disponga de la fecha y hora actualizada, es decir, la hora actual y el día de hoy. Ejecute la orden touch sobre dicho archivo y observe qué sucede sobre la fecha del citado archivo cuando se vuelva a listar.**

- Al hacer touch sobre un archivo ya existente se modifica su fecha y hora a la fecha y hora actual

**Ejercicio 2.5. La organización del espacio en directorios es fundamental para poder localizar fácilmente aquello que estemos buscando. En ese sentido, realice las siguientes acciones dentro de su directorio home (es el directorio por defecto sobre el que trabajamos al entrar en el sistema):**

a) Obtenga en nombre de camino absoluto (pathname absoluto) de su directorio home. ¿Es el mismo que el de su compañero/a?

- Nos es el mismo que el de mi compañero, en mi caso es: /home/josemhv

b) Cree un directorio para cada asignatura en la que se van a realizar prácticas de laboratorio y, dentro de cada directorio, nuevos directorios para cada una de las prácticas realizadas hasta el momento.

- Para crear estos directorios lo haremos con la orden mkdir seguido del nombre de los directorios.

c) Dentro del directorio de la asignatura fundamentos del software (llamado FS) y dentro del directorio creado para esta práctica, copie los archivos hosts y passwd que se encuentran dentro del directorio /etc.

- \$ cp -a /etc/hosts -a /etc/passwd ~/Escuela/FS/Practicas

d) Muestre el contenido de cada uno de los archivos.

- Para mostrar el contenido de cada uno de los archivos utilizamos la orden cat, por ejemplo: \$ cat hosts o \$ cat passwd

**Ejercicio 2.6. Situados en algún lugar de su directorio principal de usuario (directorío HOME), cree los directorios siguientes: Sesión.1, Sesión.10, Sesión.2, Sesión.3, Sesión.4, Sesión.27, Prueba.1 y Sintaxis.2 y realice las siguientes tareas:**

a) Borre el directorio Sesión.4

- \$ rm -d Sesión.4 , ya que el directorio está vacío, utilizamos la función -d

b) Liste todos aquellos directorios que empiecen por Sesión. y a continuación tenga un único carácter

- \$ ls Sesión.?

c) Liste aquellos directorios cuyos nombres terminen en .1

- \$ ls \*.1

d) Liste aquellos directorios cuyos nombres terminen en .1 o .2

- \$ ls -d \*.[12]

e) Liste aquellos directorios cuyos nombres contengan los caracteres si

- \$ ls \*si\*

f) Liste aquellos directorios cuyos nombres contengan los caracteres si y terminen en .2  
- \$ ls -d \*si\*.2

**Ejercicio 2.7. Desplacémonos hasta el directorio /bin, genere los siguientes listados de archivos (siempre de la forma más compacta y utilizando los metacaracteres apropiados):**

a) Todos los archivos que contengan sólo cuatro caracteres en su nombre.  
- ls -a ????

b) Todos los archivos que comiencen por los caracteres d, f.  
- \$ ls -a d\* f\*

c) Todos los archivos que comiencen por las parejas de caracteres sa, se, ad.  
- \$ ls -a sa\* se\* ad\*

d) Todos los archivos que comiencen por t y acaben en r.  
- \$ ls -a t\*r

**Ejercicio 2.8. Liste todos los archivos que comiencen por tem y terminen por .gz o .zip :**

a) De tu directorio HOME.  
- \$ ls tem\*.gz tem\*.zip

b) Del directorio actual.

c) ¿Hay alguna diferencia en el resultado de su ejecución? Razoné la respuesta.

**Ejercicio 2.9. Muestre del contenido de un archivo regular que contenga texto:**

a) Las 10 primeras líneas.  
- \$ head archivo.txt

b) Las 5 últimas líneas.  
- tail -5 archivo.txt

**Ejercicio 2.10. Cree un archivo empleando para ello cualquier editor de textos y escriba en el mismo varias palabras en diferentes líneas. A continuación trate de mostrar su contenido de manera ordenada empleando diversos criterios de ordenación.**

- \$ sort hola.txt

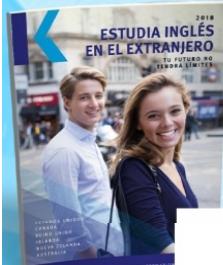
Ordena las lineas del documento situando en primer lugar las que empiezan por numeros y despues ordenandolas por orden alfabetico segun empiece cada linea

- \$ sort -r hola.txt

Ordena las lineas del documento totalmente al contraria que el caso anterior

**Ejercicio 2.11. ¿Cómo podría ver el contenido de todos los archivos del directorio actual que terminen en .txt o .c?**

- \$ cat \*.txt \*.c



# CURSOS DE INGLÉS EN EL EXTRANJERO

TU FUTURO NO TENDRÁ LÍMITES

DESCARGA  
EL CATÁLOGO  
GRATUITO

KAPLAN  
INTERNATIONAL  
ENGLISH

KAPLANINTERNATIONAL.COM/ES

41 ESCUELAS  
ALREDEDOR  
DEL MUNDO

80 AÑOS DE  
EXPERIENCIA

TODOS LOS  
NIVELES Y  
OBJETIVOS

DESCARGA  
EL CATÁLOGO  
GRATUITO



KAPLANINTERNATIONAL.COM/ES

## SESION 3

**Ejercicio 3.1.** Se debe utilizar solamente una vez la orden chmod en cada apartado. Los cambios se harán en un archivo concreto del directorio de trabajo (salvo que se indique otra cosa). Cambiaremos uno o varios permisos en cada apartado (independientemente de que el archivo ya tenga o no dichos permisos) y comprobaremos que funciona correctamente:

- Dar permiso de ejecución al “resto de usuarios”.  
\$ chmod o+x archivo

- Dar permiso de escritura y ejecución al “grupo”.  
\$ chmod g+wx archivo

- Quitar el permiso de lectura al “grupo” y al “resto de usuarios”.  
\$ chmod go-r archivo

- Dar permiso de ejecución al “propietario” y permiso de escritura el “resto de usuarios”.  
\$ chmod u+x, o+w archivo

- Dar permiso de ejecución al “grupo” de todos los archivos cuyo nombre comience con la letra “e”. Nota: Si no hay más de dos archivos que cumplan esa condición, se deberán crear archivos que empiecen con “e” y/o modificar el nombre de archivos ya existentes para que cumplan esa condición.  
\$ chmod g+x e\*

**Ejercicio 3.2.** Utilizando solamente las órdenes de la práctica anterior y los metacaracteres de redirección de salida:

- Cree un archivo llamado ej31 , que contendrá el nombre de los archivos del directorio padre del directorio de trabajo.  
\$ ls .. > ej31

- Cree un archivo llamado ej32 , que contendrá las dos últimas líneas del archivo creado en el ejercicio anterior.  
\$ tail -2 ej31 > ej32

- Añada al final del archivo ej32 , el contenido del archivo ej31 .  
\$ cat ej32 >> ej31

**Ejercicio 3.3.** Utilizando el metacarácter de creación de cauces y sin utilizar la orden cd:

- Muestre por pantalla el listado (en formato largo) de los últimos 6 archivos del directorio /etc.  
\$ ls -l /etc | tail -6

- La orden wc muestra por pantalla el número de líneas, palabras y bytes de un archivo (consulta la orden man para conocer más sobre ella). Utilizando dicha orden, muestre por pantalla el número de caracteres (sólo ese número) de los archivos del directorio de trabajo que comiencen por los caracteres “e” o “f”.  
\$ cat e\* f\* | wc -m

KAPLAN  
INTERNATIONAL  
ENGLISH

Tu nueva cuenta móvil, pensada para ti, en 8 minutos y listo.

#### **Ejercicio 3.4. Resuelva cada uno de los siguientes apartados.**

a) Cree un archivo llamado ejercicio1, que contenga las 17 últimas líneas del texto que proporciona la orden man para la orden chmod (se debe hacer en unaúnica línea de órdenes y sin utilizar el metacarácter “;” ).

```
$ man chmod | tail -17 > ejercicio1
```

b) Al final del archivo ejercicio1, añada la ruta completa del directorio de trabajo actual.

```
$ pwd >> ejercicio1
```

c)Usando la combinación de órdenes mediante paréntesis, cree un archivo llamado ejercicio3 que contendrá el listado de usuarios conectados al sistema (orden who) y la lista de archivos del directorio actual.

```
$ (who ; pwd) > ejercicio3
```

d)Añada, al final del archivo ejercicio3, el número de líneas, palabras y caracteres del archivo ejercicio1. Asegúrese de que, por ejemplo, si no existiera ejercicio1, los mensajes de error también se añadieran al final de ejercicio3.

```
$ wc ejercicio1 >> ejercicio3 >> ejercicio3
```

e)Con una sola orden chmod, cambie los permisos de los archivos ejercicio1 y ejercicio3, de forma que se quite el permiso de lectura al “grupo” y se dé permiso de ejecución a las tres categorías de usuarios.

```
$ chmod g-r,a+x ejercicio1 ejercicio3
```

#### **SESION 4**

#### **Ejercicio 4.1 Escriba, al menos, cinco variables de entorno junto con el valor que tienen.**

TERM\_PROGRAM=linux\_Terminal

TERM=xterm-256color

SHELL=/bin/bash

CLICOLOR=1

HOME=/Users/josemhvAir

#### **Ejercicio 4.2 Ejecute las órdenes del cuadro e indique qué ocurre y cómo puede resolver la situación para que la variable NOMBRE se reconozca en el shell hijo.**

```
$ NOMBRE=FS
```

```
$ echo $NOMBRE
```

```
$ bash
```

```
$ echo $NOMBRE
```

\$ NOMBRE solo es visible en el shell actual. Al cambiar el shell no podemos acceder a la variable anteriormente definida. Para exportar una variable a otro shell utilizamos las orden \$ export variable

#### **Ejercicio 4.3 Compruebe qué ocurre en las expresiones del ejemplo si se quitan las comillas dobles del final y se ponen después de los dos puntos. ¿Qué sucede si se sustituyen las comillas dobles por comillas simples?**

```
$ echo "Los archivos que hay en el directorio son: $(ls -l)"
```

```
$ echo "Los archivos que hay en el directorio son: `ls -l`"
```

Si pones las dobles comillas despues de los dos puntos, la orden sigue funcionando pero se vera la frase del echo seguida de la lista de los archivos y directorios de esa ruta pero uno detras de otros. Sustituir las comillas dobles por las simples no funciona, el echo muestra tambien la orden \$(ls -l).

**Ejercicio 4.4 Pruebe la siguiente asignación:**

```
$ numero=$numero+1
```

```
$ echo $numero
```

**¿Que ha ocurrido?**

El valor que se le asigna a la variables es esta cadena de caracteres: +1

**Ejercicio 4.5 Construya un guion que acepte como argumento una cadena de texto (por ejemplo, su nombre) y que visualice en pantalla la frase Hola y el nombre dado como argumento.**

```
#!/bin/bash
printf "Me llamo $1 \n"
```

Y para ejecutarlo usaria ./script josemhv

**Ejercicio 4.6 Varíe el guion anterior para que admita una lista de nombres.**

```
#!/bin/bash
printf "Mis amigos son $1 $2 $3 $4 \n"
Y para ejecutarlo usaria ./script Pepe Luis Paco Maria
```

**Ejercicio 4.7 Cree tres variables llamadas VAR1, VAR2 y VAR3 con los siguientes valores respectivamente “hola”, “adios” y “14”.**

a) Imprima los valores de las tres variables en tres columnas con 15 caracteres de ancho.  
printf "%-15s %-15s %-15d\n" \$VAR1 \$VAR2 \$VAR3

b) ¿Son variables locales o globales?

Son variables locales

c) Borre la variable VAR2.

```
unset VAR2
```

d) Abra otra ventana de tipo terminal, ¿puede visualizar las dos variables restantes?  
No

e) Cree una variable de tipo vector con los valores iniciales de las tres variables.  
VAR4=(hola adios 14)

f) Muestre el segundo elemento del vector creado en el apartado e.  
echo \${VAR4[1]}

**Ejercicio 4.8 Cree un alias que se llame ne (nombrado así para indicar el número de elementos) y que devuelva el número de archivos que existen en el directorio actual. ¿Qué cambiaría si queremos que haga lo mismo pero en el directorio home correspondiente al usuario que lo ejecuta?**

```
alias ne='ls -A | wc -l'
```



# CURSOS DE INGLÉS EN EL EXTRANJERO

TU FUTURO NO TENDRÁ LÍMITES

DESCARGA  
EL CATÁLOGO  
GRATUITO

KAPLAN  
INTERNATIONAL  
ENGLISH

KAPLANINTERNATIONAL.COM/ES

- 41 ESCUELAS ALREDEDOR DEL MUNDO
- 80 AÑOS DE EXPERIENCIA
- TODOS LOS NIVELES Y OBJETIVOS

DESCARGA  
EL CATÁLOGO  
GRATUITO



KAPLANINTERNATIONAL.COM/ES

Si queremos que lo haga en el directorio home:  
alias ne='cd;ls -A | wc -l'

**Ejercicio 4.9** Indique la línea de orden necesaria para buscar todos los archivos a partir del directorio home de usuario (\$HOME) que tengan un tamaño menor de un bloque. ¿Cómo la modificaría para que además imprima el resultado en un archivo que se cree dentro del directorio donde nos encontramos y que se llame archivosP?

find \$HOME -size -1

Para que imprima el resultado de la orden en un archivo llamado archivosP:  
find \$HOME -size -1 > archivosP

**Ejercicio 4.10** Indique cómo buscaría todos aquellos archivos del directorio actual que contengan la palabra “ejemplo”.  
grep ejemplo \*

**Ejercicio 4.11** Complete la información de find y grep utilizando para ello la orden man.

**Ejercicio 4.12** Indique cómo buscaría si un usuario dispone de una cuenta en el sistema.  
cat /etc/passwd | cut -d: -f1 | grep nombreUsuario

**Ejercicio 4.13** Indique cómo contabilizar el número de ficheros de la propia cuenta de usuario que no tengan permiso de lectura para el resto de usuarios.

#!/bin/bash

```
num_arch=`ls -lR $HOME|cut -d " " -f 1|grep -e '[-w][ -x]$'|wc -l`  
echo "Hay $num_arch archivos que no tienen permiso de lectura para el resto de usuarios en todo tu arbol de directorios"
```

**Ejercicio 4.14** Modifique el ejercicio 8 de forma que, en vez de un alias, sea un guion llamado numE el que devuelva el número de archivos que existen en el directorio que se le pase como argumento.

```
#!/bin/bash  
# Titulo: ejercicio 4.14  
# Autor: Jose Manuel Herrera Vera  
# Versión: 1.0  
cd;  
ls -A | wc -l;
```



# CURSOS DE INGLÉS EN EL EXTRANJERO

TU FUTURO NO TENDRÁ LÍMITES

DESCARGA  
EL CATÁLOGO  
GRATUITO

KAPLAN  
INTERNATIONAL  
ENGLISH

KAPLANINTERNATIONAL.COM/ES

41 ESCUELAS  
ALREDEDOR  
DEL MUNDO

80 AÑOS DE  
EXPERIENCIA

TODOS LOS  
NIVELES Y  
OBJETIVOS

DESCARGA  
EL CATÁLOGO  
GRATUITO



KAPLANINTERNATIONAL.COM/ES

## SESION 5

**Ejercicio 5.1:** Utilizando una variable que contenga el valor entero 365 y otra que guarde el número del día actual del año en curso, realice la misma operación del ejemplo anterior usando cada una de las diversas formas de cálculo comentadas hasta el momento, es decir, utilizando `expr`, `$(( ... ))` y `$[ ... ]`.

Primera forma:

```
$ anio=365  
$ dia=`date +%j`  
$ var=`expr $anio - $dia`  
$ echo "Faltan `expr $var / 7` semanas hasta el fin de año"
```

Segunda forma:

```
$ echo "Faltan $(( ($anio - $dia) / 7 )) semanas hasta el fin de año"
```

Tercera forma:

```
$ echo "Faltan $[ ($anio - $dia) / 7 ] semanas hasta el fin de año"
```

Combinación de ellas:

```
$ echo "Faltan `expr $[$anio - $dia] / 7` semanas hasta el fin de año"
```

**Ejercicio 5.2:** Realice las siguientes operaciones para conocer el funcionamiento del operador de incremento como sufijo y como prefijo. Razone el resultado obtenido en cada una de ellas:

**Ejercicio 5.3:** Utilizando el operador de división, ponga un caso concreto donde se aprecie que la asignación abreviada es equivalente a la asignación completa, es decir, que `x/=y` equivale a `x=x/y`.

```
x=25  
y=5  
echo $[x /= $y]  
echo $x //valdrá 5
```

**Ejercicio 5.4:** Compruebe qué ocurre si en el ejemplo anterior utiliza comillas dobles o simples para acotar todo lo que sigue a la orden `echo`. ¿Qué sucede si se acota entre comillas dobles solamente la expresión aritmética que se quiere calcular?, ¿y si se usan comillas simples?

```
echo 6/5|bc -l // muestra 1.20000000000  
echo '6/5|bc -l' // muestra literalmente lo que hay entrecomillado  
echo "6/5|bc -l" // muestra literalmente lo que hay entrecomillado  
echo '6/5'|bc -l // muestra 1.20000000000  
echo "6/5"|bc -l // muestra 1.20000000000
```

KAPLAN  
INTERNATIONAL  
ENGLISH

Tu nueva cuenta móvil, pensada para ti, en 8 minutos y listo.

**Ejercicio 5.5:** Calcule con decimales el resultado de la expresión aritmética (3-2)/5. Escriba todas las expresiones que haya probado hasta dar con una respuesta válida. Utilizando una solución válida, compruebe qué sucede cuando la expresión aritmética se acota entre comillas dobles; ¿qué ocurre si se usan comillas simples?, ¿y si se ponen apóstrofes inversos?

```
echo (3-2)/5|bc -l //Error sintactico
echo '(3-2)/5'|bc -l //Muestra .2000000000
echo "(3-2)/5"|bc -l //Muestra .2000000000
echo `^3-2)/5`|bc -l //Error sintactico, lo entrecomillado no es comando
```

**Ejercicio 5.6:** Consulte la sintaxis completa de la orden let utilizando la orden de ayuda para las órdenes empotradas (help let) y tome nota de su sintaxis general.

```
help let
```

**Ejercicio 5.7:** Con la orden let es posible realizar asignaciones múltiples y utilizar operadores que nosotros no hemos mencionado anteriormente. Ponga un ejemplo de asignación múltiple y, por otra parte, copie en un archivo el orden en el que se evalúan los operadores que admite. Apóyese a través de la ayuda que ofrece help let.

**Ejercicio 5.8:** Haciendo uso de las órdenes conocidas hasta el momento, construya un guion que admita dos parámetros, que compare por separado si el primer parámetro que se le pasa es igual al segundo, o es menor, o es mayor, y que informe tanto del valor de cada uno de los parámetros como del resultado de cada una de las evaluaciones mostrando un 0 o un 1 según corresponda.

```
#!/bin/bash

if [ $1 -eq 1 ]
    then
        echo "La primera expresion es cierta: $1"
    else
        echo "La primera expresion es falsa: $1"
    fi

if [ $2 -eq 1 ]
    then
        echo "La segunda expresion es cierta: $2"
    else
        echo "La segunda expresion es falsa: $2"
    fi

if [ $1 -lt $2 ]
    then
        echo "El primer parametro es mas pequeño que el segundo"
elif [ $1 -gt $2 ]
    then
        echo "El primer parametro es mas grande que el segundo"
else
    echo "Son iguales"
fi
```

**Ejercicio 5.9: Usando test, construya un guion que admita como parámetro un nombre de archivo y realice las siguientes acciones: asignar a una variable el resultado de comprobar si el archivo dado como parámetro es plano y tiene permiso de ejecución sobre él; asignar a otra variable el resultado de comprobar si el archivo es un enlace simbólico; mostrar el valor de las dos variables anteriores con un mensaje que aclare su significado. Pruebe el guion ejecutándolo con /bin/cat y también con /bin/rm.**

```
#!/bin/bash

planoX=`test -f $1 && test -x $1 && echo true || echo false`

simbolico=`test -h $1 && echo true || echo false`

if [ $planoX == "true" ]
then
    echo "El archivo es plano y tiene permisos de ejecucion"
else
    echo "El archivo no es plano o no tiene permisos de ejecucion"
fi

if [ $simbolico == "true" ]
then
    echo "El archivo es simbolico"
else
    echo "El archivo no es simbolico"
fi
```

**Ejercicio 5.10: Ejecute help test y anote qué otros operadores se pueden utilizar con la orden test y para qué sirven. Ponga un ejemplo de uso de la orden test comparando dos expresiones aritméticas y otro comparando dos cadenas de caracteres.**

Pruebas con cadenas:

```
$ test hola = hola
$ echo $?
$ 0          #Verdadero
$ test hola = adios
$ echo $?
$ 1          #Falso
```

Pruebas con expresiones numericas:

```
$ test $[6-3] -eq $[10-7]
$ echo $?
$ 0          #Verdadero 3 = 3
$ test $[6-3] -eq $[10+7]
$ echo $?
$ 1          #Falso 3 /= 17
```

**Ejercicio 5.11: Responda a los siguientes apartados:**

a) Razone qué hace la siguiente orden:

```
if test -f ./sesion5.pdf ; then printf "El archivo ./sesion5.pdf existe\n"; fi
```

- Verifica si el archivo "sesion5.pdf" del directorio actual es un archivo plano, si es así imprime que el archivo existe.

b) Añada los cambios necesarios en la orden anterior para que también muestre un mensaje de aviso en caso de no existir el archivo. (Recuerde que, para escribir de forma legible una orden que ocupe más de una línea, puede utilizar el carácter "\n" como final de cada línea que no sea la última.)

```
-if test -f ./sesion5.pdf ; then printf "El archivo ./sesion5.pdf existe\n"; else printf "No existe"; fi
```

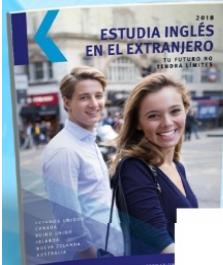
c) Sobre la solución anterior, añada un bloque elif para que, cuando no exista el archivo ./sesion5.pdf, compruebe si el archivo /bin es un directorio. Ponga los mensajes adecuados para conocer el resultado en cada caso posible.

```
-if test -f ./sesion5.pdf && test -d /bin ; then printf "El archivo ./sesion5.pdf existe y /bin es un directorio"; elif test -f ./sesion5.pdf && ! test -d /bin ; then printf "El archivo ./sesion5.pdf existe y /bin no es un directorio"; elif ! test -f ./sesion5.pdf && test -d /bin ; then printf "No existe el archivo ./sesion5.pdf y /bin es un directorio"; else printf "Ni existe ./sesion5.pdf ni /bin es un directorio"; fi
```

d) Usando como base la solución del apartado anterior, construya un guion que sea capaz de hacer lo mismo pero admitiendo como parámetros la ruta relativa del primer archivo a buscar y la ruta absoluta del segundo. Pruébelo con los dos archivos del apartado anterior.

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

```
if test -f $1 && test -d $2
    then printf "El archivo $1 existe y $2 es un directorio\n"
elif test -f $1 && ! test -d $2
    then printf "El archivo $1 existe y $2 no es un directorio\n"
elif ! test -f $1 && test -d $2
    then printf "No existe el archivo $1 y $2 es un directorio\n"
else
    printf "Ni existe $1 ni $2 es un directorio\n"
fi
```



# CURSOS DE INGLÉS EN EL EXTRANJERO

TU FUTURO NO TENDRÁ LÍMITES

DESCARGA  
EL CATÁLOGO  
GRATUITO

KAPLAN  
INTERNATIONAL  
ENGLISH

KAPLANINTERNATIONAL.COM/ES

✓ 41 ESCUELAS  
ALREDEDOR  
DEL MUNDO

✓ 80 AÑOS DE  
EXPERIENCIA

✓ TODOS LOS  
NIVELES Y  
OBJETIVOS

DESCARGA  
EL CATÁLOGO  
GRATUITO



KAPLANINTERNATIONAL.COM/ES

**Ejercicio 5.12:** Construya un guion que admita como argumento el nombre de un archivo o directorio y que permita saber si somos el propietario del archivo y si tenemos permiso de lectura sobre él.

```
#!/bin/bash

propietario=`ls -lF /bin/ls|cut -d " " -f 3`
lectura=`ls -lF /bin/ls|grep -e '^r'|wc -l`

if [ $lectura -eq 1 ]
    then
        printf "El usuario tiene permiso de lectura\n"
else
    printf "El usuario NO tiene permiso de lectura\n"
fi

if [ $1 == $propietario ]
    then
        printf "El usuario es propietario de /bin/ls\n"
else
    printf "El usuario NO es propietario de /bin/ls\n"
fi
```

**Ejercicio 5.13:** Escriba un guión que calcule si el número de días que faltan hasta fin de año es múltiplo de cinco o no, y que comunique el resultado de la evaluación. Modifique el guión anterior para que admita la opción -h de manera que, al ejecutarlo con esa opción, muestre información de para qué sirve el guión y cómo debe ejecutarse.

```
#!/bin/bash

if [[ $1 == "-h" ]]
    then
        printf "Este es el manual\n"
else
    anio=365
    dia=`date +%j`
    restantes=`expr $anio - $dia`
    printf "Faltan $restantes dias hasta el fin de año\n"

    if [ `expr $restantes % 5` == 0 ]
        then
            printf "Ademas, ese numero es multiplo de 5\n"
        else
            printf "Pero ese numero no es multiplo de 5\n"
        fi
    fi
```

**Ejercicio 5.14:** ¿Qué pasa en el ejemplo anterior si eliminamos la redirección de la orden if?

Si se borra la redirección pasa que el mensaje de error que emite el sistema operativo , por ejemplo si el archivo no existe, se muestra en pantalla además de nuestro mensaje. Si no lo eliminamos no se muestra, ya que se envía a /dev/null.

KAPLAN  
INTERNATIONAL  
ENGLISH

Tu nueva cuenta móvil, pensada para ti, en 8 minutos y listo.

**Ejercicio 5.15: Haciendo uso del mecanismo de cauces y de la orden echo, construya un guión que admita un argumento y que informe si el argumento dado es una única letra, en mayúsculas o en minúsculas, o es algo distinto de una única letra.**

```
#!/bin/bash

if [ `echo $1 | grep -e '^.$` ]
then
    echo "Es un unico caracter"

    if [[ $1 =~ [A-Z] ]]
    then
        echo "Y es una letra mayuscula"
    elif [[ $1 =~ [a-z] ]]
    then
        echo "Y es una letra minuscula"
    else
        echo "No es una letra, es otro caracter"
    fi

else
    echo "No es un unico caracter"
fi
```

**Ejercicio 5.16: Haciendo uso de expresiones regulares, escriba una orden que permita buscar en el árbol de directorios los nombres de los archivos que contengan al menos un dígito y la letra e. ¿Cómo sería la orden si quisieramos obtener los nombres de los archivos que tengan la letra e y no contengan ni el 0 ni el 1?**

```
find / -regex '.*\([0-9]+.*e\).*'
find . -regex '.*e.*' -and -not -regex '.*[01].*' 
```

**Ejercicio 5.17: Utilizando la orden grep, exprese una forma alternativa de realizar lo mismo que con wc -l.**

```
grep -c
```

## COMANDOS DE UNIX::SESION 2

<b>Órdenes</b>	<b>Descripción</b>
<b>ls [directorio]</b>	Lista los contenidos de un directorio
<b>cd [directorio]</b>	Cambia de directorio de trabajo. Las abreviaciones . y .. se pueden utilizar como referencia de los directorios actual y padre, respectivamente. El símbolo ~ (pulsando a la vez las teclas AltGr y 4) es el directorio HOME del usuario y el símbolo / al inicio de un camino es el directorio raíz del sistema.
<b>pwd</b>	Imprime el camino absoluto del directorio actual
<b>mkdir directorio</b>	Crea un directorio a partir del nombre dado como argumento
<b>rmdir directorio</b>	Borra un directorio existente (si está vacío)
<b>cat [archivo(s)]</b>	Orden multipropósito: muestra el contenido de un archivo o varios, concatena archivos, copia un archivo, crea un archivo de texto o muestra los caracteres invisibles de control.
<b>cp archivo1 archivo2</b>	Copia el archivo1 en el archivo2. Si archivo2 no existe, se crea.

<b>mv fuente destino</b>	Renombra archivos (el archivo fuente puede ser archivo o directorio, al igual que el destino) y puede mover de lugar un archivo o directorio
<b>file archivo(s)</b>	Muestra el tipo de archivo dado como argumento
<b>more archivo(s)</b>	Visualiza un archivo fraccionándolo una pantalla cada vez (existen otros paginadores como page, pg, etc.). Antes de usar esta orden es conveniente usar la orden file para comprobar que se trata de un archivo ascii.
<b>rm directorio_archivos</b>	Borra archivos y directorios con contenido
<b>touch archivo(s)</b>	Si existen los archivos dados como argumentos se modifican su fecha y hora. En caso contrario, se crean con la fecha actual del sistema.
<b>clear</b>	Borra el contenido del terminal actual
<b>tail [archivo(s)]</b>	Muestra la parte final del contenido de un archivo dado como argumento. Por defecto muestra 10 líneas.
<b>head [archivo(s)]</b>	Muestra la parte inicial del contenido de un archivo dado como argumento. Por defecto muestra 10 líneas.
<b>sort [archivo(s)]</b>	Ordena, según un criterio elegido, el contenido de los archivos dados como argumentos



# CURSOS DE INGLÉS EN EL EXTRANJERO

TU FUTURO NO TENDRÁ LÍMITES

DESCARGA  
EL CATÁLOGO  
GRATUITO

KAPLAN  
INTERNATIONAL  
ENGLISH

KAPLANINTERNATIONAL.COM/ES

- ✓ 41 ESCUELAS ALREDEDOR DEL MUNDO
- ✓ 80 AÑOS DE EXPERIENCIA
- ✓ TODOS LOS NIVELES Y OBJETIVOS

DESCARGA  
EL CATÁLOGO  
GRATUITO



KAPLANINTERNATIONAL.COM/ES

## OPCIONES DE LOS COMANDOS:

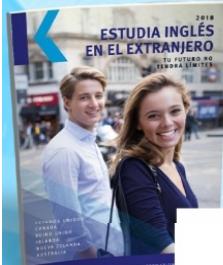
### ls: opciones mas importantes:

- ls -l → muestra el contenido de un directorio pero además muestra los permisos de cada archivo de ese directorio.
- ls -a → todos los archivos, incluso los que comienzan con punto (.) .
- ls -A → Lista todos los ficheros en los directorios, excepto los que comienzan con punto . (.) y los que comienzan con doble punto (..).
- ls -F → indica tipo: / directorio, \* ejecutable, @ enlace simbólico.
- ls -h → indicará el tamaño en KB, MB, etc.
- ls -l → listado en formato largo (o detallado).
- ls -S → clasifica los contenidos de los directorios por tamaños, con los ficheros más grandes en primer lugar.
- ls -r → invierte el orden de la salida.
- ls -R → Lista recursivamente los subdirectorios encontrados.
- ls -t → ordenar por fecha de última modificación.
- ls -u → ordenar por fecha de último acceso.
- ls -x → presenta los ficheros por columnas.
- ls -i → precede la salida con el número de **i-node** (ver el comando [ln](#)).

### cd uso de algunas abreviaciones:

- cd directorio/directorio/todos los directorios que queramos → uso común
- cd .. → volver al directorio anterior
- cd ~/directorio → Éste comando irá al directorio de inicio del usuario que es "/home/username" y tras / ponemos el directorio al que queremos acceder.

Ejemplo:: cd ~/Escritorio/ejemplo1



# CURSOS DE INGLÉS EN EL EXTRANJERO

TU FUTURO NO TENDRÁ LÍMITES

DESCARGA  
EL CATÁLOGO  
GRATUITO

KAPLAN  
INTERNATIONAL  
ENGLISH

KAPLANINTERNATIONAL.COM/ES

✓ 41 ESCUELAS  
ALREDEDOR  
DEL MUNDO

✓ 80 AÑOS DE  
EXPERIENCIA

✓ TODOS LOS  
NIVELES Y  
OBJETIVOS

DESCARGA  
EL CATÁLOGO  
GRATUITO

KAPLANINTERNATIONAL.COM/ES

## cat opciones:

- A Mostrar todo.
- b Omitir los números de línea para los espacios en blanco en la salida.
- e Un carácter \$ se mostrará al final de cada línea anterior a una nueva línea.
- E Muestra un \$ (símbolo del dolar) al final de cada línea.
- n Numera todas las líneas en el salida.
- s Si el salida tiene múltiples líneas vacías las sustituye con una única línea vacía.
- T Muestra los caracteres de tabulación en el salida.
- v Los caracteres no mostrados (con la excepción de tabuladores, nuevas líneas y saltos de página) se muestran.

## Ejemplos:

1. Para crear un archivo nuevo:

`cat > file1.txt`

Este comando crea un archivo nuevo file1.txt. Tras escribir en el archivo presiona control+d (^d) simultáneamente para finalizar el archivo.

2. Para añadir información al archivo:

`cat >> file1.txt`

Para añadir información en el mismo archivo utiliza el operador de adición >> para escribir en el archivo, si no, el archivo será sobreescrito (todo su contenido será eliminado).

3. Para mostrar un archivo:

`cat file1.txt`

Este comando muestra la información en el archivo.

4. Para concatenar varios archivos y mostrarlos:

`cat file1.txt file2.txt`

El comando cat anterior concatenará los dos archivos (file1.txt y file2.txt) y mostrará el salida en la pantalla. Algunas veces el salida no cabrá dentro de la pantalla. En dicha situación puedes mostrar esos archivos en un archivo nuevo o mostrar el archivo utilizando el comando less.

`cat file1.txt file2.txt |less`

5. Para concatenar varios archivos y transferir el salida a otro archivo.

`cat file1.txt file2.txt > file3.txt`

KAPLAN  
INTERNATIONAL  
ENGLISH

Tu nueva cuenta móvil, pensada para ti, en 8 minutos y listo.

En el ejemplo anterior la salida se redirige al nuevo archivo file3.txt. El comando cat creará el nuevo archivo file3.txt y guardará el salida concatenado en file3.txt.

### **cp opciones mas relevantes:**

- f → Borrar los archivos de destino ya existentes.
- d → Copiar los enlaces simbólicos tal cual son, en lugar de copiar los archivos a los que apuntan.
- p → Preservar los permisos, el usuario y el grupo del archivo a copiar.
- R → Copiar directorios recursivamente.
- a → Equivalente a utilizar las opciones -dpR.
- u → No copia un archivo si en el destino ya existe tal archivo, y éste tiene la fecha de modificación igual o mas reciente.
- v → Da información en pantalla sobre los archivos que se van copiando.

Los permisos se indican de la siguiente forma:

Permiso	Archivos	Directorios
r	Lectura (read)	Se puede listar su contenido
w	Escritura (write)	Podemos modificarlo
x	Ejecución (execute)	Podemos acceder a él
-	No hay permiso	

Existen tres grupos de permisos:

rwX	rwX	rwx
Propietario del archivo (user)	Grupo de usuarios (group)	Resto de usuarios (others)

Metacarácter	Descripción de la función
?	Representa cualquier carácter simple en la posición en la que se indique
*	Representa cualquier secuencia de cero o más caracteres
[ ]	Designan un carácter o rango de caracteres que representan un carácter simple a través de una lista de caracteres o mediante un rango, en cuyo caso, mostramos el primer y último carácter del rango separados por un guión "-".
{ }	Sustituyen conjuntos de palabras separadas por comas que comparten partes comunes.
~	Se usa para abreviar el camino absoluto (path) del directorio HOME.

## COMANDOS DE UNIX::SESION 3

<b>ÓRDENES</b>	<b>DESCRIPCIÓN</b>
chmod <grupo+ó-permisos> archivo	Asigna o quita algún tipo de permisos a algún grupo
wc archivo	Muestra por pantalla el número de líneas, palabras y bytes de un archivo
echo “mostrado en pantalla”	Muestra en pantalla lo que escribamos entre comillas dobles
date	Proporciona la fecha y hora del sistema

<b>Metacarácter</b>	<b>Descripción</b>
< <b>nombre</b>	Redirecciona la entrada de una orden para que la obtenga del archivo <i>nombre</i> .
> <b>nombre</b>	Redirige la salida de una orden para que la escriba en el archivo <i>nombre</i> . Si dicho archivo ya existe, lo sobreescribe.
&> <b>nombre</b>	La salida estándar se combina con la salida de error estándar y ambas se escriben en el archivo <i>nombre</i> .
>> <b>nombre</b>	Funciona igual que el metacarácter ">" pero añade la salida estándar al final del contenido del archivo <i>nombre</i> .
&>> <b>nombre</b>	Igual que el metacarácter ">>", pero añadiendo las dos salidas combinadas al final del archivo <i>nombre</i> .
2> <b>nombre</b>	Redirige la salida de error estándar a un archivo (sólo funciona en shells de "bash").
	Crea un cauce entre dos órdenes. La salida de una de ellas se utiliza como entrada de la otra.
&	Crea un cauce entre dos órdenes utilizando las dos salidas (estándar y error) de una de ellas como entrada de la otra.

<b>Metacarácter</b>	<b>Descripción</b>
:	Separador entre órdenes que se ejecutan secuencialmente.
( )	Se usan para aislar órdenes separadas por ";" o por " ". Las órdenes dentro de los paréntesis son tratadas como una única orden.
&&	Separador entre órdenes, en la que la orden que sigue al metacarácter "&&" se ejecuta sólo si la orden precedente ha tenido éxito (no ha habido errores).
	Separador entre órdenes, en la que la orden que sigue al metacarácter "  " se ejecuta sólo si la orden precedente falla.

**OPCIONES DE LOS COMANDOS:****chmod uso:**

- chmod u+rwx archivo → da permisos de lectura, escritura y ejecución al propietario.
- chmod g+rwx archivo → da permisos de lectura, escritura y ejecución a los grupos de usuarios.
- chmod o+rwx archivo → da permisos de lectura, escritura y ejecución al resto de usuarios.
- chmod u-x archivo → quita permiso de ejecución al propietario
- chmod -R a+rwx → da permisos a todos los usuarios a todos los directorios y ficheros recursivamente.

Se podrían hacer muchas combinaciones mas con los visto anteriormente , lo de antes simplemente eran unos cuantos ejemplos.

**wc uso:**

- wc -c → Contar bytes.
- wc -l → Contar líneas.
- wc -w → Contar palabras.
- wc -m → Contar caracteres.
- wc -L → Longitud de la linea mas larga.

**echo opcion -e:**

- \n para hacer un intro:

```
$ echo -e 'ejemplo\nintro'
ejemplo
intro
```

- \t para un tabulador:

```
$ echo -e 'tab\ttab'
tab      tab
```

- \v para un "tabulador vertical", lo que también se conoce como un salto de linea (sin retorno de carro):

```
$ echo -e 'lol\vlol'
lol
lol
```

- \b para backspace, se desplaza al carácter anterior:

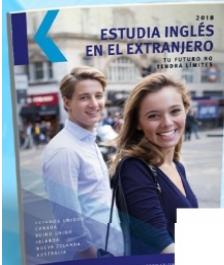
```
$ echo -e 'a\bb'
b
```

Resulta muy útil para hacer dibujos, como la siguiente barra giratoria ASCII:

```
$ while true; do for i in / - \\ '|'; do echo -n $i; sleep 1; done; done
```

- \| por si queremos usar la misma contrabarra:

```
$ echo -e '\\'
\
```



# CURSOS DE INGLÉS EN EL EXTRANJERO

TU FUTURO NO TENDRÁ LÍMITES

DESCARGA  
EL CATÁLOGO  
GRATUITO

KAPLAN  
INTERNATIONAL  
ENGLISH

KAPLANINTERNATIONAL.COM/ES

✓ 41 ESCUELAS  
ALREDEDOR  
DEL MUNDO

✓ 80 AÑOS DE  
EXPERIENCIA

✓ TODOS LOS  
NIVELES Y  
OBJETIVOS

DESCARGA  
EL CATÁLOGO  
GRATUITO



KAPLANINTERNATIONAL.COM/ES

## date:

- date +%a → Día de la semana abreviado.
- date +%A → Día de la semana completo.
- date +%b → Nombre del mes abreviado.
- date +%B → Nombre del mes completo.
- date +%d → Día del mes.
- date +%m → Número de mes.
- date +%H → Hora, en formato 24h.
- date +%M → Minutos.
- date +%S → Segundos.

Ejemplo:: date +"%A %d %B"

## COMANDOS DE UNIX::SESION 4

ÓRDENES	DESCRIPCIÓN
set variable=valor	Declara variables locales y sin argumentos muestra las variables locales.
unset variable	Elimina variables locales.
env	Visualiza las variables de entorno.
printenv	Visualiza las variables de entorno.
declare -opción variable	Declara variables locales diciendo el tipo de variable en la opción
expr expresión matemática	Resuelve cálculos matemáticos
export variable=valor	Convierte las variables locales en variables globales o de entorno.
alias mote='comandos'	Asigna un “mote” a unas ordenes
unalias mote	Borra el “mote”
find lista-de-directorios expresiones	Buscar por la estructura de directorios los archivos que satisfagan los criterios especificados.
grep opciones patrón archivos	Permite buscar cadenas en archivos utilizando patrones para especificar dicha cadena.
fgrep opciones patrón archivos	Acepta sólo una cadena simple de búsqueda en vez de una expresión regular.
egrep opciones patrón archivos	Permite un conjunto más complejo de operadores en expresiones regulares.

KAPLAN  
INTERNATIONAL  
ENGLISH

Tu nueva cuenta móvil, pensada para ti, en 8 minutos y listo.

printf formato argumentos	Imprime un mensaje en la pantalla utilizando el formato que se le especifica.
<b>Nombre de variable</b>	<b>Descripción</b>
\$BASH	Contiene la ruta de acceso completa usada para ejecutar la instancia actual de bash.
\$HOME	Almacena el directorio raíz del usuario; se puede emplear junto con la orden cd sin argumentos para ir al directorio raíz del usuario.
\$PATH	Guarda el camino de búsqueda de las órdenes, este camino está formado por una lista de todos los directorios en los que queremos buscar una orden.
\$?	Contiene el código de retorno de la última orden ejecutada, bien sea una instrucción o un guion.

### 4.3 La orden empotrada printf

Nombre	Descripción
\$0	Nombre del guion o script que se ha llamado. Sólo se emplea dentro del guion.
\$1 ... \$9 \${n}, n>9	Son los distintos argumentos que se pueden facilitar al llamar a un guion. Los nueve primeros se referencian con \$1, \$2, ..., \$9, y a partir de ahí es necesario encerrar el número entre llaves, es decir, \${n}, para n>9.
\$*	Contiene el nombre del guion y todos los argumentos que se le han dado. Cuando va entre comillas dobles es equivalente a "\$1 \$2 \$3 ... \$n".
\$@	Contiene el nombre del guion y todos los argumentos que se le han dado. Cuando va entre comillas dobles es equivalente a "\$1" "\$2" ... "\$n".
\$#	Contiene el número de argumentos que se han pasado al llamar al guion.
\${arg:-val}	Si el argumento tiene valor y es no nulo, continua con su valor, en caso contrario se le asigna el valor indicado por val.
\${arg:?val}	Si el argumento tiene valor y es no nulo, sustituye a su valor; en caso contrario, imprime el valor de val y sale del guion. Si val es omitida, imprime un mensaje indicando que el argumento es nulo o no está asignado.

Tabla 4.5. Variables de entorno definidas para los argumentos de un guion.

Código de formato	Representa
%d	Un número con signo
%f	Un número en coma flotante (decimal) sin notación exponencial
%q	Entrecomilla una cadena
%s	Muestra una cadena sin entrecomillar
%x	Muestra un número en hexadecimal
%o	Muestra un número en octal

Tabla 4.4. Algunos códigos de formato.

## **OPCIONES DE LOS COMANDOS:**

### **set:**

- **set -a** → A partir de este momento, las variables que se declaran se exportan automáticamente a globales sin necesidad de ejecutar "export variable" .
- **set -b** → Si disponemos de una aplicación que se ejecuta en segundo plano, nos reporta el estado una vez haya terminado.
- **set -vx** → Muestra las líneas que va ejecutando nuestra script. Para desactivar-lo: set +xv .
- **set variable=valor** → Se crea una variable local .

### **declare:**

- declare -i variable → variable numérica.
- declare -r variable → indica que es de solo lectura.
- declare -a variable → indica que es una matriz.
- declare -x variable → indicará que es exportable.
- declare -p variable → ver atributos.

### **find:**



# CURSOS DE INGLÉS EN EL EXTRANJERO

TU FUTURO NO TENDRÁ LÍMITES

DESCARGA  
EL CATÁLOGO  
GRATUITO

KAPLAN  
INTERNATIONAL  
ENGLISH

KAPLANINTERNATIONAL.COM/ES

✓ 41 ESCUELAS ALREDEDOR DEL MUNDO

✓ 80 AÑOS DE EXPERIENCIA

✓ TODOS LOS NIVELES Y OBJETIVOS

DESCARGA  
EL CATÁLOGO  
GRATUITO



KAPLANINTERNATIONAL.COM/ES

**-print:** visualiza los nombres de camino de cada archivo que se adapta al criterio de búsqueda. Es la opción por defecto. Por ejemplo, para visualizar los nombres de todos los archivos y directorios del directorio actual:

```
$ find . -print
```

**-exec:** permite añadir una orden que se aplicará a los archivos localizados. La orden se situará a continuación de la opción y debe terminarse con un espacio, un carácter \ y a continuación un ;. Se utiliza {} para representar el nombre de archivos localizados. Por ejemplo:

```
$ find . -atime +100 -exec rm {} \;
```

eliminará todos los archivos del directorio actual (y sus descendientes) que no ha sido utilizados en los últimos 100 días.

**-ok:** es similar a **-exec**, con la excepción de que solicita confirmación en cada archivo localizado antes de ejecutar la orden.

```
$ find / -name "*.c"
```

2. Por el último acceso: se utiliza la opción **-atime** seguida por un número de días o por el número y un signo + o - delante de él. Por ejemplo:

**-atime 7** busca los archivos a los que se accedió hace 7 días.

**-atime -2** busca los archivos a los que se accedió hace menos de 2 días.

**-atime +5** busca los archivos a los que se accedió hace más de 5 días.

3. Por ser de un determinado tipo: se utiliza la opción **-type** seguida de un carácter que indique el tipo de archivo. Se usa la opción f para referirse a archivos regulares y la opción d para directorios. En el ejemplo siguiente se muestra la búsqueda de los archivos del directorio actual que sean archivos regulares:

```
$ find . -type f
```

4. Por su tamaño en bloques: se utiliza la opción **-size** seguida de un número con o sin signo (+ o -). Si el número va seguido de la letra c el tamaño dado es en bytes. Por ejemplo: **-size 100** busca los archivos cuyo tamaño es de 100 bloques.

Además, se puede negar cualquier operador de selección o de acción utilizando el operador ! que debe ir entre espacios en blanco y antes del operador a negar. Por ejemplo, para buscar los archivos del directorio raíz que no pertenezcan al usuario llamado pat:

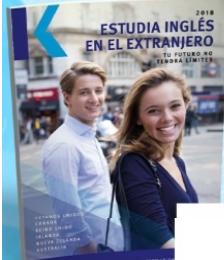
```
$ find / ! -user pat
```

También se puede especificar un operador u otro utilizando el operador **-o**. Este operador conecta dos expresiones y se seleccionarán aquellos archivos que cumplan una de las dos expresiones (y no las dos, como hasta ahora). En el siguiente ejemplo se muestra la búsqueda de los archivos de tamaño igual a 10 bloques o cuyo último acceso (modificación) se haya efectuado hace más de dos días:

```
$ find . -size 10 -o -atime +2
```

Las acciones más comunes son:

**grep:**



# CURSOS DE INGLÉS EN EL EXTRANJERO

TU FUTURO NO TENDRÁ LÍMITES

DESCARGA  
EL CATÁLOGO  
GRATUITO

KAPLAN  
INTERNATIONAL  
ENGLISH

KAPLANINTERNATIONAL.COM/ES

41 ESCUELAS  
ALREDEDOR  
DEL MUNDO

80 AÑOS DE  
EXPERIENCIA

TODOS LOS  
NIVELES Y  
OBJETIVOS

DESCARGA  
EL CATÁLOGO  
GRATUITO



KAPLANINTERNATIONAL.COM/ES

## 4.5.2 Órdenes grep, egrep y fgrep

La orden `grep` permite buscar cadenas en archivos utilizando patrones para especificar dicha cadena. Esta orden lee de la entrada estándar o de una lista de archivos especificados como argumentos y escribe en la salida estándar aquellas líneas que contengan la cadena. Su formato es:

`grep opciones patrón archivos`

En su forma más sencilla, el patrón puede ser una cadena de caracteres, aunque, como se verá en la siguiente práctica, también pueden usarse expresiones regulares. Por ejemplo, para buscar la palabra `mundo` en todos los archivos del directorio actual usaremos:

```
$ grep mundo *
```

Algunas opciones que se pueden utilizar con la orden `grep` son:

- x localiza líneas que coincidan totalmente, desde el principio hasta el final de línea, con el patrón especificado.
- v selecciona todas las líneas que no contengan el patrón especificado.
- c produce solamente un recuento de las líneas coincidentes.
- i ignora las distinciones entre mayúsculas y minúsculas.
- n añade el número de línea en el archivo fuente a la salida de las coincidencias.
- l selecciona sólo los nombres de aquellos archivos que coincidan con el patrón de búsqueda.
- e especial para el uso de múltiples patrones e incluso si el patrón comienza por el carácter (-).

Existen dos variantes de `grep` que optimizan su funcionamiento en casos especiales. La orden `fgrep` acepta sólo una cadena simple de búsqueda en vez de una expresión regular. La orden `egrep` permite un conjunto más complejo de operadores en expresiones regulares. Usando `man` comprueba las diferencias entre estas tres órdenes.

## COMANDOS DE UNIX::SESION 5

KAPLAN  
INTERNATIONAL  
ENGLISH

Tu nueva cuenta móvil, pensada para ti, en 8 minutos y listo.

ÓRDENES	DESCRIPCIÓN
((...))	Evaluá una expresión aritmética y sustituye el resultado de la expresión en el lugar donde se utiliza.
[...]	Evaluá una expresión aritmética y sustituye el resultado de la expresión en el lugar donde se utiliza.
bc -opción expresión_matemática	Operaciones con decimales.
let variable.entera=expresión_matemática	Evaluar expresiones aritméticas.
test	Evaluá una expresión condicional y da como salida el estado 0, en caso de que expresión se haya evaluado como verdadera ( true ), o el estado 1, si la evaluación ha resultado falsa ( false ) o se le dio algún argumento no válido.
if/else	Ejecuta una lista de declaraciones dependiendo de si se cumple o no cierta condición.

Operador	Descripción
+ -	Suma y resta, o más unario y menos unario.
* / %	Multiplicación, división (truncando decimales), y resto de la división.
**	Potencia.
++	Incremento en una unidad. Puede ir como prefijo o como sufijo de una variable: si se usa como prefijo ( <code>++variable</code> ), primero se incrementa la variable y luego se hace lo que se desee con ella; si se utiliza como sufijo ( <code>variable++</code> ), primero se hace lo que se desee con la variable y después se incrementa su valor.
--	Decremento en una unidad. Actúa de forma análoga al caso anterior, pudiendo usarse como prefijo o como sufijo de una variable ( <code>--variable</code> o <code>variable--</code> ).
( )	Agrupación para evaluar conjuntamente; permite indicar el orden en el que se evaluarán las subexpresiones o partes de una expresión.
,	Separador entre expresiones con evaluación secuencial.
=	x=expresión, asigna a x el resultado de evaluar la expresión (no puede haber huecos en blanco a los lados del símbolo "=");
+= -=	$x+=y$ equivale a $x=x+y$ ; $x-=y$ equivale a $x=x-y$ ;
*=/	$x*=y$ equivale a $x=x*y$ ; $x/=y$ equivale a $x=x/y$ ;
%=	$x\%=y$ equivale a $x=x \% y$ .

**Tabla 5.2.** Operadores aritméticos.

Operador			Descripción: el resultado se evalúa como "verdadero" - <i>true</i> - si ... (en otro caso sería "falso" - <i>false</i> )
A = B	A == B	A -eq B	A es igual a B.
A != B		A -ne B	A es distinta de B.
A < B		A -lt B	A es menor que B.
A > B		A -gt B	A es mayor que B.
A <= B		A -le B	A es menor o igual que B.
A >= B		A -ge B	A es mayor o igual que B.
! A			A es falsa; representa al operador NOT (negación lógica).
A && B			A es verdadera y B es verdadera; es el operador AND (conjunción lógica).
A    B			A es verdadera o B es verdadera; es el operador OR (disyunción lógica).

**Tabla 5.3.** Operadores relacionales.

Expresiones regulares con órdenes de búsqueda(find,grep,egrep):

Patrón	Representa
\	la barra de escape; si en un patrón se quiere hacer referencia a este mismo carácter, debe ir precedido por él mismo y ambos entre comillas simples.
.	cualquier carácter en la posición en la que se encuentre el punto cuando se usa en un patrón con otras cosas; si se usa solo, representa a cualquier cadena; si se quiere buscar un punto como parte de un patrón, debe utilizarse \. entre comillas simples o dobles.
( )	un grupo; los caracteres que se pongan entre los paréntesis serán considerados conjuntamente como si fuesen un único carácter. (Hay que usar \)
?	que el carácter o grupo al que sigue puede aparecer una vez o no aparecer ninguna vez. (Hay que usar \)
*	que el carácter o grupo al que sigue puede no aparecer o aparecer varias veces seguidas. (No hay que usar \)
+	que el carácter o grupo previo debe aparecer una o más veces seguidas.
{n}	que el carácter o grupo previo debe aparecer exactamente n veces. (Hay que usar \)
{n,}	que el carácter o grupo previo debe aparecer n veces o más seguidas. (Hay que usar \)
{n,m}	que el carácter o grupo previo debe aparecer de n a m veces seguidas; al menos n veces, pero no más de m veces. (Hay que usar \)
[ ]	una lista de caracteres que se tratan uno a uno como caracteres simples; si el primer carácter de la lista es "^", entonces representa a cualquier carácter que no esté en esa lista.

-	un rango de caracteres cuando el guion no es el primero o el último en una lista; si el guion aparece el primero o el último de la lista, entonces se trata como él mismo, no como rango; en los rangos de caracteres, el orden es el alfabetico, pero intercalando minúsculas y mayúsculas – es decir: aAbB...; en los rangos de dígitos el orden es 012... También es posible describir rangos parciales omitiendo el inicio o el final del rango (por ejemplo [m-] representa el rango que va desde la "m" hasta la "z").
^	indica el inicio de una línea; como se ha dicho anteriormente, cuando se usa al comienzo de una lista entre corchetes, representa a los caracteres que no están en esa lista. Situando a continuación de ^ un carácter, filtrará todas aquellas líneas que comiencen por ese carácter.
\$	indica el final de una línea. Situando un carácter antes del \$, filtrará todas aquellas líneas que terminen por ese carácter.
\b	el final de una palabra. (Debe utilizarse entre comillas simples o dobles)
\B	que no está al final de una palabra. (Debe utilizarse entre comillas simples o dobles)
\<	el comienzo de una palabra. (Debe utilizarse entre comillas simples o dobles)
\>	el final de una palabra. (Debe utilizarse entre comillas simples o dobles)
\	el operador OR para unir dos expresiones regulares, de forma que la expresión regular resultante representa a cualquier cadena que coincida con al menos una de las dos subexpresiones. (La expresión global debe ir entre comillas simples o dobles; además, cuando se usa con grep, esta orden debe ir acompañada de la opción -E)

**Tabla 5.5.** Operadores para expresiones regulares.

## OPCIONES DE LOS COMANDOS:

### bc:

- bc -l → Selecciona la biblioteca matemática estándar.(recomendada siempre )

### let:

```
$ let w=3+2
$ let w='3 + 2'
$ let w='(4+5)*6'
$ let "w=4+5*6"
$ let w=4+5*6
$ y=7
$ let w=y%5
                (esta orden es equivalente a: let w=$y%5)
```

Como habrá observado en el ejemplo anterior, las dos primeras asignaciones producen el mismo resultado, a pesar de que en la segunda hay espacios en blanco. Por el contrario, las asignaciones tercera y cuarta no dan el mismo resultado debido al uso o no de paréntesis. Las asignaciones cuarta y quinta son equivalentes, y las dos últimas ponen de manifiesto que en la expresión pueden intervenir otras variables.

### test:

- sintaxis: test -opcion(mirar tabla 5.4) archivo.
- <test expresion> == [ expresion ]

Operador	Descripción: el resultado se evalúa como "verdadero" - <i>true</i> - si ... (en otro caso sería "falso" - <i>false</i> -)
-a archivo	archivo existe.
-b archivo	archivo existe y es un dispositivo de bloques.
-c archivo	archivo existe y es un dispositivo de caracteres.
-d archivo	archivo existe y es un directorio.
-e archivo	archivo existe. Es igual que -a.
-f archivo	archivo existe y es un archivo plano o regular.
-G archivo	archivo existe y es propiedad del mismo grupo del usuario.
-h archivo	archivo existe y es un enlace simbólico.
-L archivo	archivo existe y es un enlace simbólico. Es igual que -h.
-O archivo	archivo existe y es propiedad del usuario.
-r archivo	archivo existe y el usuario tiene permiso de lectura sobre él.
-s archivo	archivo existe y es no vacío.
-w archivo	archivo existe y el usuario tiene permiso de escritura sobre él.
-x archivo	archivo existe y el usuario tiene permiso de ejecución sobre él, o es un directorio y el usuario tiene permiso de búsqueda en él.
archivol -nt archivo2	archivol es más reciente que archivo2, según la fecha de modificación, o si archivol existe y archivo2 no.
archivol -ot archivo2	archivol es más antiguo que archivo2, según la fecha de modificación, o si archivo2 existe y archivol no.
archivol -ef archivo2	archivol es un enlace duro al archivo2, es decir, si ambos se refieren a los mismos números de dispositivo e inode.

Tabla 5.4. Operadores de consulta de archivos.



# CURSOS DE INGLÉS EN EL EXTRANJERO

TU FUTURO NO TENDRÁ LÍMITES

DESCARGA  
EL CATÁLOGO  
GRATUITO

KAPLAN  
INTERNATIONAL  
ENGLISH

KAPLANINTERNATIONAL.COM/ES

✓ 41 ESCUELAS  
ALREDEDOR  
DEL MUNDO

✓ 80 AÑOS DE  
EXPERIENCIA

✓ TODOS LOS  
NIVELES Y  
OBJETIVOS

DESCARGA  
EL CATÁLOGO  
GRATUITO



KAPLANINTERNATIONAL.COM/ES

## if/else:

- sintaxis:

```
if condición;  
then  
    declaraciones;  
[elif condición;  
    then declaraciones; ]...  
[else  
    declaraciones; ]  
fi
```

condición : test expresión ó [ ... ]

KAPLAN  
INTERNATIONAL  
ENGLISH

Tu nueva cuenta móvil, pensada para ti, en 8 minutos y listo.