WUOLAH

jsuallo
www.wuolah.com/student/jsuallo

PracticasFSModI.pdf

Prácticas FS Módulo 1

1° Fundamentos del Software

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Relación de prácticas Fundamentos del Software

Curso 2019-20

Grado en Ingeniería Informática. Grupo A3

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

INEAF
BUSINESS SCHOOL

Es el momento
DE CRECER

Master en Asesoría Fiscal de Empresas

Módulo I

Práctica 1

Ejercicio 1.1. 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 ejercicio1
$ mkdir ejercicio1/Ejer1
$ mkdir ejercicio1/Ejer2
$ mkdir ejercicio1/Ejer3
$ mkdir 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.

```
$ cd Ejer21
$ touch robet202.me
$ touch ejerllsol.txt
$ touch blue.me
```

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

```
$ cd ../../Ejer2
$ touch ejer2arch.txt
$ touch ejer2filetags.txt
$ touch readme2.pdf
```

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

```
$ cd ../Ejer3
$ touch ejer3arch.txt
$ touch ejer3filetags.txt
$ touch readme3.pdf
```

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

```
$ mkdir -p ejercicio/Ejer{1,2,3} ejercicio1/Ejer1/Ejer21
```

Ejercicio 1.2. Situados en el directorio ejercicio1 creado anteriormente, realice las siguientes acciones:

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

a) Mueva el directorio Ejer21 al directorio Ejer2.

```
$ mv Ejer1/Ejer21 Ejer2/
```

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

```
$ cp Ejer1/*.*x* Ejer3/
```

c) Si estamos situado en el directorio Ejer2 y ejecutamos la orden ls -la ../Ejer3/*arch*, ¿qué archivo/s, en su caso, debería mostrar?

Debería mostrar los archivos “ejer1arch.txt” y “ejer3arch.txt”.

Ejercicio 1.3. 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 1.4. 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.

```
$ ls –l
-rwx--x--x 1 javsuallo javsuallo 287 Oct 23 16:46 4_10
-rw-r--r-- 1 javsuallo javsuallo  14 Oct 23 16:46 4_12
-rwx------ 1 javsuallo javsuallo 346 Oct 23 16:46 4_13
-rwxr--r-- 1 javsuallo javsuallo 240 Oct 24 11:50 4_14_1
-rwxr-xr-x 1 javsuallo javsuallo 241 Oct 23 16:46 4₉
$ touch 4_10
$ ls –l
-rwx--x--x 1 javsuallo javsuallo 287 Oct 27 22:31 4_10
-rw-r--r-- 1 javsuallo javsuallo  14 Oct 23 16:46 4_12
-rwx------ 1 javsuallo javsuallo 346 Oct 23 16:46 4_13
-rwxr--r-- 1 javsuallo javsuallo 240 Oct 24 11:50 4_14_1
-rwxr-xr-x 1 javsuallo javsuallo 241 Oct 23 16:46 4_9
```

Ejercicio 1.5. 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?

```
$ pwd ../
```

/home | **/home**

Si, es el mismo camino.

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

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.

```
$ mkdir FP
$ mkdir FS
$ mkdir ALEM
$ mkdir CA
$ mkdir FFT
$ mkdir FP/S1
$ mkdir CA/S1
$ mkdir FS/S1
```

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 /etc/passwd ./
$ cp /etc/hosts ./
```

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

```
$ cat FS/S1/hosts
$ cat FS/S1/passwd
```

Ejercicio 1.6. Situados en algún lugar de su directorio principal de usuario (directorio HOME), cree los directorios siguientes: Sesion.1, Sesion.10, Sesion.2, Sesion.3, Sesion.4, Sesion.27, Prueba.1 y Sintaxis.2 y realice las siguientes tareas:

a) Borre el directorio Sesion.4

```
$ rm Sesion.4
```

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

```
$ ls Sesion.?
```

c) Liste aquellos directorios cuyos nombres terminen en .1

```
$ ls *.1
```

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

```
$ ls *.{1,2}
```

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 *si*.2
```

Formación Online Especializada

Clases Online
Prácticas
Becas

Escuela de LÍDERES

Rafael García Parrado
Director de Marketing Digital

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Ejercicio 1.7. 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 ????
```

b) Todos los archivos que comiencen por los caracteres d, f.

```
$ ls d* f*
```

c) Todos los archivos que comiencen por las parejas de caracteres sa, se, ad.

```
$ ls sa* se* ad*
```

d) Todos los archivos que comiencen por t y acaben en r.

```
$ ls t*r
```

Ejercicio 1.8. Liste todos los archivos que comiencen por tem y terminen por .gz o .zip :

a) De tu directorio HOME.

```
$ ls /home/ tem*.{gz, zip}
```

b) Del directorio actual.

```
$ ls tem*.{gz, zip}
```

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

Sí hay diferencia, ya que en uno se hace un listado en /home y el otro en el directorio /bin, directorio en el que estábamos trabajando en el ejercicio 1.7.

Ejercicio 1.9. Muestre del contenido de un archivo regular que contenga texto:

a) Las 10 primeras líneas.

```
$ head [archivo]
```

b) Las 5 últimas líneas.

```
$ tail -n 5 [archivo]
```

Ejercicio 1.10. 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 -d [archivo]
$ sort -R [archivo]
$ sort -r [archivo]
$ sort -b [archivo]
```

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

Ejercicio 1.11. ¿Cómo podría ver el contenido de todos los archivos del directorio actual que terminen en .txt o .c?

```
$ ls *.txt *.c
```

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

Práctica 2

Ejercicio 2.1. 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".

```
-rw-r--r-- 1 jsuallo alumno 0 sep 30 10:36 e1
$ chmod o+x e1
$ ls -l e1
-rw-r--r-x 1 jsuallo alumno 0 sep 30 10:36 e1
```

- Dar permiso de escritura y ejecución al "grupo".

```
-rw-r--r-- 1 jsuallo alumno 0 sep 30 10:36 e2
$ chmod g+wx e2
-rw-rwxr-- 1 jsuallo alumno 0 sep 30 10:36 e2
```

- Quitar el permiso de lectura al "grupo" y al "resto de usuarios".

```
-rw-r--r-- 1 jsuallo alumno 0 sep 30 10:36 e3
$ chmod og-r e3
-rw------- 1 jsuallo alumno 0 sep 30 10:36 e3
```

- Dar permiso de ejecución al "propietario" y permiso de escritura el "resto de usuarios".

```
-rw-r--r-- 1 jsuallo alumno 0 sep 30 10:36 e4
$ chmod u+x,o+w e4
-rwxr--rw- 1 jsuallo alumno 0 sep 30 10:36 e4
```

- Dar permiso de ejecución al "grupo" de todos los archivos cuyo nombre comience con la letra "e".

```
-rw-r--r-x 1 jsuallo alumno 0 sep 30 10:36 e1
-rw-rwxr-- 1 jsuallo alumno 0 sep 30 10:36 e2
-rw------- 1 jsuallo alumno 0 sep 30 10:36 e3
-rwxr--rw- 1 jsuallo alumno 0 sep 30 10:36 e4
$ chmod g+x e*
-rw-r-xr-x 1 jsuallo alumno 0 sep 30 10:36 e1
-rw-rwxr-- 1 jsuallo alumno 0 sep 30 10:36 e2
-rw---x--- 1 jsuallo alumno 0 sep 30 10:36 e3
-rwxr-xrw- 1 jsuallo alumno 0 sep 30 10:36 e4
```

(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.)

Ejercicio 2.2. Utilizando solamente las órdenes de la práctica anterior y los metacaracteres de redirección de salida:

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

- 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.

```
$ cat ej31 | tail –2 ej31 > ej32
```

- Añada al final del archivo ej32 , el contenido del archivo ej31.

```
$ cat ej32 >> ej31
```

Ejercicio 2.3. 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 /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”.

```
$ wc –m e* f*
```

Ejercicio 2.4. 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 una única 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; ls) > 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
```

Formación Online Especializada

Clases Online
Prácticas
Becas

Escuela de LÍDERES

Rafael García Parrado
Director de Marketing Digital

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

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
```

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

Práctica 3

Los ejercicios **3.1** – **3.4** están hechos en clase.

Ejercicio 3.5. 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
echo “Hola $1”
```

Ejercicio 3.6. Varíe el guion anterior para que admita una lista de nombres.

```
#!/bin/bash
echo “Hola $*”
```

Ejercicio 3.7. Cree tres variables llamadas VAR1, VAR2 y VAR3 con los siguientes valores respectivamente “hola”, “adios” y “14”.

```
VAR1=hola
VAR2=adios
VAR3=14
```

a) Imprima los valores de las tres variables en tres columnas con 15 caracteres de ancho.

```
printf "%15q" $VAR1 $VAR2 $VAR3
```

b) ¿Son variables locales o globales?

Son variables de tipo local.

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.

```
set VAR=(hola adios 14)
```

f) Muestre el segundo elemento del vector creado en el apartado e.

Ejercicio 3.8. 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=`find ./ | wc –l`
```

```
$ alias ne=`find .. | wc –l`
```

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

Ejercicio 3.9. 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 encontremos y que se llame archivosP?

```
$ find .. -size 1b
```

```
$ find .. -size 1b > ./archivosP
```

Ejercicio 3.10. Indique cómo buscaría todos aquellos archivos del directorio actual que contengan la palabra “ejemplo”.

```
$ find .. *ejemplo*
```

Ejercicio 3.11. Complete la información de find y grep utilizando para ello la orden man.

```
$ man find
```

```
$ man grep
```

Ejercicio 3.12. Indique cómo buscaría si un usuario dispone de una cuenta en el sistema.

```
$ find /home [usuario]
```

Ejercicio 3.13. Indique cómo contabilizar el número de ficheros de la propia cuenta de usuario que no tengan permiso de lectura para el resto de usuarios.

```
$ find .. -perm /o+r
```

Ejercicio 3.14. 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

archivos=`find $1 | wc -l`

printf "Existen %d archivos en el directorio %q.\n" $archivos $1
```

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

Práctica 4

Ejercicio 4.1: 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 $[...].

```
$ diasanio=365
$ diahoy=`date +%j`
$ echo $(`expr $diasanio - $diahoy`)
$ echo $(( $diasanio - $diahoy ))
```

Ejercicio 4.3: 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=4
$ y=2
$ echo $[ x/=y ]
$ echo $[ x=x/y ]
```

Ejercicio 4.4: 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?

Si utilizamos comillas dobles, se imprime el resultado de la expresión x/=y; y si utilizamos comillas simples imprime "$[x/=y]".

Ejercicio 4.5: 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 apóstrofos inversos?

```
$ echo $[(3-2)]/5|bc -l
```

Con comillas dobles, la orden funciona de la misma manera. Con apóstrofos inversos la orden da error.

Ejercicio 4.6: 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.

```
let: let arg [arg ...]
    Evalúa expresiones aritméticas.

     Evalúa cada ARG como una expresión aritmética.  La evaluación se
hace
    con enteros de longitud fija, sin revisar desbordamientos, aunque la
    la división por 0 se captura y se marca como un error.  La siguiente
     lista de operadores está agrupada en niveles de operadores de la
misma prioridad.Se muestran los niveles en orden de prioridad
decreciente.

     id++, id-- post-incremento, post-decremento de variable
     ++id, --id pre-incremento, pre-decremento de variable
     -, +       menos, más unario
     !, ~       negación lógica y basada en bits
```

Formación Online Especializada

Clases Online
Prácticas
Becas

Escuela de LÍDERES

Rafael García Parrado
Director de Marketing Digital

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

```
      **          exponenciación
      *, /, %           multiplicación, división, residuo
      +, -        adición, sustracción
      <<, >>            desplazamientos de bits izquierdo y derecho
      <=, >=, <, >      comparación
      ==, !=            equivalencia, inequivalencia
      &           AND de bits
      ^           XOR de bits
      |           OR de bits
      &&          AND lógico
      ||          OR lógico
      expr ? expr : expr
                  operador condicional
      =, *=, /=, %=,
      +=, -=, <<=, >>=,
      &=, ^=, |= asignación

    Se permiten las variables de shell como operandos.  Se reemplaza
    el nombre de la variable por su valor (coercionado a un entero
    de longitud fija) dentro de una expresión.  La variable no
    necesita tener activado su atributo integer para ser usada en
    una expresión.

    Los operadores se evalúan en orden de prioridad.  Primero se
    evalúan las sub-expresiones en paréntesis y pueden sobrepasar
    las reglas de prioridad anteriores.

    Estado de salida:
    Si el último ARGumento se evalúa como 0, let devuelve 1; de otra
    forma, let devuelve 0.
```

Ejercicio 4.7: 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.

```
$ let w=x=5
```

Ejercicio 4.8: Probad los siguientes ejemplos y escribir los resultados obtenidos con la evaluación de expresiones

```
echo ejemplo1
valor=6
if [ $valor = 3 ]; then echo si; else echo no; fi
si
echo $valor
6
```

```
echo ejemplo2
valor=5
if [ $valor = 3 ] && ls; then echo si; else echo no; fi
no
echo $valor
5
```

```
echo ejemplo3
valor=5
if [ $valor = 5 ] && ls; then echo si; else echo no; fi
Descargas   Escritorio     Música         Público
Documentos  Imágenes       Plantillas     Vídeos
si
```

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

```
echo $valor
5
```

```
echo ejemplo4
valor=6
if ((valor==3)); then echo si; else echo no; fi
echo $valor
```

```
echo ejemplo5
valor=5
if ((valor==3)) && ls; then echo si; else echo no; fi
echo $valor
```

```
echo ejemplo6
valor=5
if ((valor==5)) && ls; then echo si; else echo no; fi
echo $valor
```

```
echo ejemplo7
echo $((3>5))
echo $?
```

```
echo ejemplo8
((3>5))
echo $?
```

```
echo ejemplo9
if ((3>5)); then echo 3 es mayor que 5; else echo 3 no es mayor que 5;
fi
```

Ejercicio 4.9: 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

x=$1
y=$2

if (($x<$y)); then printf "El primer valor es menor que el segundo.\n";
fi

if (($x>$y)); then printf "El segundo valor es menor que el primero.\n";
fi

if (($x==$y)); then printf "Ambos valores son iguales.\n"; fi
```

Ejercicio 4.10: 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/rnano.

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

```
#!/bin/bash

archivo=$1

existe_plano=`test -f $archivo -a -x $1 && echo "Sí" || echo "No"`
enlace_simb=`test -h $archivo && echo "Sí" || echo "No"`


printf "Análisis del archivo $archivo:\n"
printf "¿Es plano? $existe_plano\n"
printf "¿Es un enlace simbólico? $enlace_simb\n"
```

Ejercicio 4.11: 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.

```
      -a FILE        True if file exists.
      -b FILE        True if file is block special.
      -c FILE        True if file is character special.
      -d FILE        True if file is a directory.
      -e FILE        True if file exists.
      -f FILE        True if file exists and is a regular file.
      -g FILE        True if file is set-group-id.
      -h FILE        True if file is a symbolic link.
      -L FILE        True if file is a symbolic link.
      -k FILE        True if file has its `sticky' bit set.
      -p FILE        True if file is a named pipe.
      -r FILE        True if file is readable by you.
      -s FILE        True if file exists and is not empty.
      -S FILE        True if file is a socket.
      -t FD          True if FD is opened on a terminal.
      -u FILE        True if the file is set-user-id.
      -w FILE        True if the file is writable by you.
      -x FILE        True if the file is executable by you.
      -O FILE        True if the file is effectively owned by you.
      -G FILE        True if the file is effectively owned by your
group.
      -N FILE        True if the file has been modified since it was
last read.

      FILE1 -nt FILE2  True if file1 is newer than file2 (according to
                       modification date).

      FILE1 -ot FILE2  True if file1 is older than file2.

      FILE1 -ef FILE2  True if file1 is a hard link to file2.

    All file operators except -h and -L are acting on the target of a
symbolic link, not on the symlink itself, if FILE is a symbolic link.

    String operators:

      -z STRING      True if string is empty.

      -n STRING
         STRING      True if string is not empty.

      STRING1 = STRING2
                     True if the strings are equal.
      STRING1 != STRING2
```

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

```
                       True if the strings are not equal.
      STRING1 < STRING2
                       True if STRING1 sorts before STRING2
lexicographically.
      STRING1 > STRING2
                       True if STRING1 sorts after STRING2
lexicographically.

    Other operators:

      -o OPTION      True if the shell option OPTION is enabled.
      -v VAR     True if the shell variable VAR is set
      -R VAR     True if the shell variable VAR is set and is a name
reference.
      ! EXPR         True if expr is false.
      EXPR1 -a EXPR2 True if both expr1 AND expr2 are true.
      EXPR1 -o EXPR2 True if either expr1 OR expr2 is true.

      arg1 OP arg2   Arithmetic tests.  OP is one of -eq, -ne,
                     -lt, -le, -gt, or -ge.
```

```
$ test num_1 < num_2
```

```
$ test str_1 < str_2
```

Ejercicio 4.12: 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

Comprueba que el archivo “sesion5.pdf” existe en el directorio actual y, si existe, imprime en la terminal “El archivo ./sesion5.pdf existe”. En caso contrario no imprime nada.

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 “\” 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 “El archivo ./sesion5.pdf no existe.\n”; 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 ; then printf “El archivo ./sesion5.pdf existe\
n”; elif test -d /bin; then printf “/bin es un directorio.\n” ; else
printf “El archivo ./sesion5.pdf no existe.\n”; fi
```

Formación Online Especializada

Clases Online
Prácticas
Becas

Escuela de LÍDERES

Rafael García Parrado
Director de Marketing Digital

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

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;
   then
   printf “El archivo %s existe\n” $1;

   elif test –d $2;
   then printf “%s es un directorio.\n” $2;
   else
   printf “El archivo %s no existe.\n” $1;
   fi
```

Ejercicio 4.13: 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

archivo=$1

prop_y_exe=`test –x $1 –a –O $archivo && echo "true" || echo "false"`

if $prop_y_exe == `true`; then printf "El archivo $archivo existe, es
propiedad del usuario y además tiene permiso de ejecución.\n"; else
printf "El archivo $archivo no existe, o no es propiedad del usuario o
no tiene permiso de ejecución.\n"; fi
```

Ejercicio 4.14: Escriba un guion 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 guion anterior para que admita la opción -h de manera que, al ejecutarlo con esa opción, muestre información de para qué sirve el guion y cómo debe ejecutarse.

```
#!/bin/bash
dias=`365 – ($date +%j)`

if dias%5 == 0; then printf “El número de días restantes hasta fin de
año es múltiplo de 5.”; else printf “El número de días restantes hasta
fin de año no es múltiplo de 5.”; fi
```

```
#!/bin/bash

if $1==’-h’;
   then

   printf “AYUDA: El programa busca un programa añadido como argumento y
comprueba si existe, es propiedad del usuario y éste tiene permiso de
ejecución\n”;

   else
   archivo=$1

   prop_y_exe=`test –x $1 –a –O $archivo && echo "true" || echo "false"`

   if $prop_y_exe == `true`; then printf "El archivo $archivo existe, es
propiedad del usuario y además tiene permiso de ejecución.\n"; else
```

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

```
printf "El archivo $archivo no existe, o no es propiedad del usuario o
no tiene permiso de ejecución.\n"; fi;
fi
```

El siguiente guion de ejemplo se puede utilizar para borrar el archivo temporal que se le dé como argumento. Si rm devuelve 0, se muestra el mensaje de confirmación del borrado; en caso contrario, se muestra el código de error. Como se puede apreciar, hemos utilizado la variable $LINENO que indica la línea actualmente en ejecución dentro del guion.

```
#!/bin/bash
declare -rx SCRIPT=${0##*/}
# donde SCRIPT contiene sólo el nombre del guión
# ${var##Patron} actúa eliminando de $var aquella parte
# que cumpla de $Patron desde el principio de $var
# En este caso: elimina todo lo que precede al
# último slash “/".

if rm ${1} ; then
printf "%s\n" "$SCRIPT: archivo temporal borrado"
else
STATUS=177
printf "%s – código de finalizacion %d\n" \
"$SCRIPT:$LINENO no es posible borrar archivo" $STATUS
fi 2> /dev/null
```

Ejercicio 4.15: ¿Qué pasa en el ejemplo anterior si eliminamos la redirección de la orden if?

Si eliminamos “if”, el guión ejecutará todas las sentencias independientemente del estado de finalización de la orden “rm”.

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

Práctica 5

Ejercicio 5.1. Escriba un guion que acepte dos argumentos. El primero será el nombre de un directorio y el segundo será un valor entero. El funcionamiento del guion será el siguiente: deberán anotarse en un archivo denominado archivosSizN.txt aquellos archivos del directorio dado como argumento y que cumplan la condición de tener un tamaño menor al valor aportado en el segundo argumento. Se deben tener en cuenta las comprobaciones sobre los argumentos, es decir, debe haber dos argumentos, el primero deberá ser un directorio existente y el segundo un valor entero.

```
#!/bin/bash

if [ $# != 2 ];
then
   printf "AYUDA: Debe introducir dos parámetros: un directorio y
un número entero.\n";
else
   esnum=`if let a -eq $2; then echo "true"; else echo "false";
fi`
   if $esnum && `test -d $1` ;
   then
      printf "El resultado se almacenará en el archivo
archivosSizN.txt\n"
      for (( CONTADOR=1; CONTADOR<=$2; CONTADOR++ )) ; do
      find $1 -size $CONTADOR >> archivosSizN.txt
      done;
   else
      printf "ERROR: Alguno de los argumentos no es válido.\n";
   fi;
fi
```

Ejercicio 5.2. Escriba un guion que acepte el nombre de un directorio como argumento y muestre como resultado el nombre de todos y cada uno de los archivos del mismo y una leyenda que diga "Directorio", "Enlace" o "Archivo regular", según corresponda. Incluya la comprobación necesaria sobre el argumento, es decir, determine si el nombre aportado se trata de un directorio existente.

```
#!/bin/bash

if [ $# != 0 ];
then
   if `test -d $1`;
   then
      printf "Directorios:\n"
      find $1 -type d
      printf "Enlaces:\n"
      find $1 -type l
      printf "Archivos regulares:\n"
      find $1 -type f;
```

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

```
   else
      printf "ERROR: El directorio no existe.\n";
   fi;
else
   printf "ERROR: No ha introducido ningún directorio.\n";
fi
```

Ejercicio 5.3. Escriba un guion en el que, a partir de la pulsación de una tecla, detecte la zona del teclado donde se encuentre. Las zonas vendrán determinadas por las filas. La fila de los números 1, 2, 3, 4, ... será la fila 1, las teclas donde se encuentra la Q, W, E, R, T, Y,... serán de la fila 2, las teclas de la A, S, D, F, ... serán de la fila 3 y las teclas de la Z, X, C, V, ... serán de la fila 4. La captura de la tecla se realizará mediante la orden read.

```
#!/bin/bash

read –p "Pulse una tecla: " INPUT

case "$INPUT" in
    1|2|3|4|5|6|7|8|9|0) printf "La tecla pertenece a la primera
fila: 1 2 3 4 5 6 7 8 9 0\n";;
    Q|W|E|R|T|Y|U|I|O|P) printf "La tecla pertenece a la segunda
fila: Q W E R T Y U I O P\n";;
    A|S|D|F|G|H|J|K|L|Ñ) printf "La tecla pertenece a la tercera
fila: A S D F G H J K L Ñ\n";;
   Z|X|C|V|B|N|M) printf "La tecla pertenece a la cuarta fila: Z X
C V B N M\n";;
   *) printf "La tecla pertenece a otra zona del teclado.\n";;
esac
```

Ejercicio 5.4. Escriba un guion que acepte como argumento un parámetro en el que el usuario indica el mes que quiere ver, ya sea en formato numérico o usando las tres primeras letras del nombre del mes, y muestre el nombre completo del mes introducido. Si el número no está comprendido entre 1 y 12 o las letras no son significativas del nombre de un mes, el guion deberá mostrar el correspondiente mensaje de error.

```
#!/bin/bash

case "$1" in
   1|'ene') printf "Enero\n";;
   2|'feb') printf "Febrero\n";;
   3|'mar') printf "Marzo\n";;
   4|'abr') printf "Abril\n";;
   5|'may') printf "Mayo\n";;
   6|'jun') printf "Junio\n";;
   7|'jul') printf "Julio\n";;
   8|'ago') printf "Agosto\n";;
   9|'sep') printf "Septiembre\n";;
   10|'oct') printf "Octubre\n";;
   11|'nov') printf "Noviembre\n";;
   12|'dic') printf "Diciembre\n";;
   *) printf "ERROR: Debe introducir un valor válido.\n";;
esac
```

Formación Online Especializada

Clases Online
Prácticas
Becas

Escuela de LÍDERES

Rafael García Parrado
Director de Marketing Digital

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Ejercicio 5.5. Escriba un guion que solicite un número hasta que su valor esté comprendido entre 1 y 10. Deberá usar la orden while y, para la captura del número, la orden read.

```
#!/bin/bash

printf "Introduzca un número entre 1 y 10.\n"
numero=0

while (( $numero < 1 || $numero > 10 )) ;
   do
   read numero;
  done
```

Los ejercicios a partir del **5.6** no se hacen.

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito

Práctica 6

Ejercicio 6.3. Escribir un guion que nos dé el nombre del proceso del sistema que consume más memoria.

```
$ ps –eo cmd,pmem ––sort pmem | tail –n 1
```

Ejercicio 6.4. Escribir un guion que escriba números desde el 1 en adelante en intervalos de un segundo ¿Cómo se podría, desde otro terminal, detener la ejecución de dicho proceso, reanudarlo y terminar definitivamente su ejecución?

```
#!/bin/bash
contador=0

while true;
do
   contador=`expr $contador + 1`
   echo "Contador vale ahora: " $contador
   sleep 1
done
```

```
$ ps –e
$ bg
$ kill –s STOP 18939
```

Ejercicio 6.5. ¿Se puede matar un proceso que se encuentra suspendido? En su caso, ¿cómo?

Con la orden kill.

Ejercicio 6.6. ¿Qué debemos hacer a la orden top para que nos muestre sólo los procesos nuestros?

```
$ top –u jsuallo
```

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Recordatorio de Wuolah: HOY SI QUE ERES UN CRACK #KeepCalm y #estudiaUnPoquito