Passos bàsics per a crear i executar un shellscript

1. Crear un fitxer de text amb un editor de text qualsevol. Es recomanable posar-hi l'extensió ***.sh**
   a. La primera línia del fitxer haurà de ser **#!/bin/bash** (conegut com a *shebang*).
   b. A continuació podeu introduir-hi totes les comandes del shellscript.
2. Donar permisos d'execució al fitxer creat amb la comanda **chmod 700 nom_fitxer.sh**
3. Executar el shellscript amb **./nom_fitxer.sh**

Definició de variables

Quan volem assignar un valor a una variable ho fem amb el signe =, sempre sense espais.

```
$ a=2; # assigna un valor individual

$ b=hola; # assigna una cadena de caràcters (sense espais)

$ c="hola que tal"; # assigna una cadena de caràcters (amb espais)
```

- Quan volem assignar un valor a una variable ho fem amb el signe **=**, sempre sense espais.

```
$ echo $c; # mostra el contingut de c

$ a=$b; # assigna b al contingut de a
```

- El shell disposa de tres metacaràcters de tipus cometa:
  - **Cometes simples (' ')**: Indiquen que cal interpretar literalement la cadena de caràcters que conté.
  - **Cometes dobles (" ")**: Indiquen que cal interpretar literalment tota la cadena de caràcters que conté, excepte el metacaràcter $ pel valor de les variables.
  - **Cometes inverses (` `)**: Indiquen que cal executar la cadena de caràcters que conté. Típicament s'utilitzen per a utilitzar el resultat d'una execució com a paràmetres d'una altra comanda.

Cometes

```
$ echo $PATH *; # mostra el PATH i el llistat del directori

$ echo '$PATH *'; # mostra la cadena literalment

$ echo "$PATH *"; # només substitueix el valor del PATH

$ ls -l `which sort`; # utilitza el resultat de which sort com a paràmetre de ls -l

$ users=`cut -d: f1 /etc/passwd`  # assigna a la variable users el contingut de la primera columna del fitxer /etc/passwd (què conté els identificadors dels usuaris del sistema).

$ echo \*; echo * # mostra com \ també inhibeix l'expansió del metacaràcter *. El seu efecte es limita a un únic caràcter
```

Comandes útils (1)

- Farem servir la comanda **echo** en diverses ocasions, que permet escriure per la sortida estàndard.

```
$ echo "Sistemes Operatius";

$ echo -e "\ta\tb"; # el paràmetre -e activa la interpretació de caràcters especials (com ara el \t, que representa el tabulador)
```

- Una altra comanda que trobarem freqüentment serà **sleep**, que permet adormir o aturar l'execució durant el nombre de segons que es passin com a paràmetre.

```
$ sleep 10
```

- **expr**: executa una expressió matemàtica i mostra el resultat per la sortida estàndard.

```
$ expr 3 + 4;  # mostra 7
$ expr 4 - 2;  # mostra 2
$ expr 3 \* 4;  # mostra 12 (cal protegir-lo)
$ expr 12 / 4;  # mostra 3
$ expr 15 % 4;  # mostra 3, el mòdul
$ a=`expr 3 + 7`; # emmagatzema 10 a la variable a
```

- **read**: carrega els continguts de l'entrada estàndard a una variable.

```
$ read valor;  # llegeix el valor introduït per teclat
$ echo $valor;  # el mostrem per pantalla
$ read lin; echo Lin:$lin # carrega la línia llegida a la variable lin i mostra el resultat
$ read word1 word2; echo L1:$word1; echo L2:$word2 # carrega la primera paraula llegida a word1; la següent paraula (i la resta, si és que n'hi ha) és guardaran a word2
```

- **test**: avalua condicions respecte un arxiu (si existeix, si és llegible, modificable, …) respecte a cadenes de caràcters (si són iguals…) o nombres (=, ≠, <, >, ≤, ≥,…).
  - No mostra res per la sortida estàndard, però té un codi de sortida estàndard (accessible des de la variable $?).
  - Retornarà **0 si és certa** i **un valor diferent de 0** en cas contrari.

```
$ test -d /bin; echo $?  # mostra 0 perquè /bin és un directori
$ test -w /bin; echo $?  # 1 perquè no podem escriure a /bin
$ test hola = adeu ; echo $?;  # mostra 1 perquè 4 no és més gran que 5
$ test 3 -ne 6; echo $? # mostra 0 perquè 3 és diferent de 6
$ test 3 -gt 2 -a 5 -lt; echo $? # mostra 0 perquè tota la condició és certa (-a = and)
```

- **printf**: mostra un missatge per la sortida estàndard. Admet especificar cadenes de format com la rutina printf de llenguatge C.

```
$ printf '%x\n' 15;  # escriu l'enter 15 en base hexadecimal (format %x)
$ printf '%10s\n' abc;  # afegeix espais en blanc a l'esquerra d'abc
```

- **exit**: Finalitza l'execució del shellscript. Si voleu, podeu especificar un paràmetre de tipus enter; aquest valor representa el codi d'acabament del shellscript i pot ser útil perquè el shellscript informi del seu resultat (de forma anàloga a la comanda test).

- **true/false**: comandes que es fan servir per generar les **condicions de control dels bucles infinits**.

Control del flux d'execució: **if**

A **condicióN** podem tenir, per exemple:

- **test**
- **grep**
  - cert si ha trobat la paraula
- el resultat de qualsevol altra comanda:
  - Totes tenen un codi de retorn que sol ser cert quan tot ha anat bé.
  - Si es comuniquen diverses comandes amb pipes, es fa servir el valor retornat per la última comanda.

```
if condició1
then
        sentències1
else if condició2
then
        sentències2
...
else
        sentènciesN
fi
```

Control del flux d'execució: **while/until**

- **while** itera mentre la condició sigui certa.
- **until** itera mentre la condició sigui falsa.

A "condició", podem tenir el mateix tipus de condicions que a **if**.

```
while condició
do
        sentències
done


until condició
do
        sentències
done
```

Control del flux d'execució: **for**

La semàntica de la sentència **for** en bash és lleugerament diferent de la de JavaScript.

La sentència **for** itera per a cadascun dels valors continguts a la llista:

- Si indiquem un filtre, iterarà sobre els fitxers que el compleixin:
  - * tots els fitxers del directori actual
  - ../src/*.c, /proc/*/cmdline, …
- Si indiquem $* iterarem sobre tots els paràmetres del shellscript.
- comanda iterarà sobre els resultats de la comanda indicada.
  - En ocasions s'utilitza la comanda **seq** (generarà seqüències de nombres).

```
for variable in llista
do
        sentències
done
```

-

Control del flux d'execució: **case**

La sentència **case** busca quin és el primer patró que compleix la paraula indicada i executa les sentències corresponents.

Dins el patró es poden utilitzar metacaràcters:

- a*: paraules que comencen per a.
- a*|b*: paraules que comencen per a o b.
- *: qualsevol (típicament l'últim patró).

El cas '*' és opcional. Representa l'opció es realitzarà quan no es compleixen cap dels patrons anteriors.

```
case paraula in
        patro1)
                sentencies1
                ;;
        patro2)
                sentencies2
                ;;
        ...
        patroN)
                sentenciesN
                ;;
        *)
                sentencies
                ;;
esac
```

Consells per a programar shellscripts

- Abans de començar a escriure el shellscript, **prova les comandes** una a una en la línia de comandes.
  - És més senzill veure que funcionen individualment i, després, unir-les per tal d'obtenir la funcionalitat desitjada.
  - No pretengueu escriure els shellscripts de cop! És millor programar per parts i anar fent proves.
- Recordeu que **l'assignació de valors a variables no porta cap espai** (=).
- **Utilitzeu xivatos** (**echo**) per a verificar valors i comportaments.
- Cal comprovar que el **nombre de paràmetres** és correcte.
  - Si no ho és, caldrà mostrar un missatge per pantalla que indiqui l'ús del programa.
-

Exercici 5

Crea un shellscript anomenat Setmana en el qual es passi per paràmetre un nombre de l'1 al 7, i segons el nombre es retorni un dia de la setmana (1: Dilluns, 2: Dimarts...7: Diumenge).

És important comprovar què el paràmetre sigui correcte i únic (només ha d'haver 1). Si no és correcte, s'ha de retornar un missatge d'error.

```
#!/bin/bash

# comprovam que el paràmetre sigui únic i un nombre de l'1 al 7
if `test $# -ne 1`
then
        echo "El nombre de paràmetres introduïts no és correcte"
        exit 1
fi

if `test $1 -lt 1 -o $1 -gt 7`
then
        echo "El nombre ha de ser d'1 a 7"
        exit 1
fi

# mostram el resultat segons el nombre introduït
case $1 in
        1)
                echo "$1: Dilluns"
        ;;
        2)
                echo "$1: Dimarts"
        ;;
        3)
                echo "$1: Dimecres"
        ;;
        4)
                echo "$1: Dijous"
        ;;
        5)
                echo "$1: Divendres"
        ;;
        6)
                echo "$1: Dissabte"
        ;;
        7)
                echo "$1: Diumenge"
        ;;
esac
```

Exercici 6

Crea un shellscript parescut a l'anterior on es puguin passar més d'un paràmetre i per cadascun s'indiqui quin dia de la setmana és.

També s'han de comprovar els paràmetres.

```
#!/bin/bash

# comprovam que el paràmetre sigui únic
if `test $# -eq 0`
then
        echo "S'ha d'introduir un o més paràmetres"
        exit 1
fi

# recorrem cada paràmetre per donar resultat
for d in "$@"
do

        # comprovam si el paràmetre és un nombre d'1 a 7
        if `test $d -lt 1 -o $d -gt 7`
        then
                echo "El nombre ha de ser d'1 a 7"
        fi

        # mostram el resultat segons el nombre introduït
        case $d in
                1)
                        echo "$d: Dilluns"
                ;;
                2)
                        echo "$d: Dimarts"
                ;;
                3)
                        echo "$d: Dimecres"
                ;;
                4)
                        echo "$d: Dijous"
                ;;
                5)
                        echo "$d: Divendres"
                ;;
                6)
                        echo "$d: Dissabte"
                ;;
                7)
                        echo "$d: Diumenge"
                ;;
                # enlloc de posar l'if per controlar si és de 1 a 7
                # també podem posar aquesta opció
                *)
                        echo "$d: ERROR S'ha d'introduir un nombre d'1 a 7"
                ;;
        esac
done
```

Exercici 2

Expliqueu el funcionament de les cometes d'aquest exercici, i el per què d'això.

```
#!/bin/bash

comanda=ls

echo “$comanda”
echo `$comanda`
echo ‘$comanda’
```

```
#!/bin/bash

comanda=ls

echo "$comanda" # Es mostrarà el valor de la variable, ls
echo `$comanda` # S'executarà el contingut de la variable, la comanda ls
echo '$comanda' # Es mostrarà literalment la cadena de caràcters, $comanda
```

Exercici 3

Indiqueu justificadament quina és la funció del següent shellscript, indicant quin és el significat més lògic dels paràmetres. Després, expliqueu una comanda equivalent a tot aquest shellscript.

```
#!/bin/bash

touch tmp

for i in *.txt
do
   grep “examen” $i >> tmp
done

wc -l < tmp
rm tmp
```

```
#!/bin/bash

# Cream un fitxer temporal
touch tmp

# Per cada fitxer .txt del directori actual, es guardaran les línies que contenguin la paraula examen al fitxer temporal, sense solapament
for i in *.txt
do
        grep "examen" $i >> tmp
done

# mostrarem el nombre de línies totals i eliminam el fitxer temporal
# es mostren el nombre de línies que contenen la paraula examen dels fitxers de text del directori actual
wc -l < tmp
rm tmp
```

Exercici 4

Escriviu un shellscript que, a partir d'un paràmetre numèric N, ordeni alfabèticament els noms dels usuaris del sistema i d'aquests mostra els N últims.

```
#!/bin/bash

# comprovam que s'hagi introduït un paràmetre, si no és així, sortim
if `test $# -ne 1`
then
        echo "ERROR: introdueix  un caràcter numèric"
        exit 1
fi

# cut-> obtenim el nom dels usuaris
# sort i tail -> els ordinam i es mostren els $1 últims
cut -d ":" -f1 /etc/passwd | sort | tail -n $1
```

Exercici 5

Escriviu un shellscript que indiqui si ens trobem als primers o als últims sis mesos de l'any. Cal que tingueu en compte que el sistema pot estar en qualsevol idioma, per tant, utilitzeu els paràmetres de la comanda date per a obtenir un valor vàlid per a qualsevol idioma.

```
#!/bin/bash

# guardam el mes actual a una variable
mes=`date +%m`

if `test $mes -le 6`
then
        echo "Primers 6 mesos de l'any"
else
        echo "Darrers 6 mesos de l'any"
fi
```

Exercici 6

Escriviu un shellsript simple que a partir d'un nombre indeterminat d'arguments, saludi a cadascun dels arguments passats. Per exemple:

```
./exercici Adrián Gerard Raúl Iker
Hola Adrián
Hola Raúl
Hola Iker
```

```
#!/bin/bash

# Comprovam que s'hagin introduit paràmetres
if `test $# -eq 0`
then
        echo "ERROR: Introdueix al menys un paràmetre"
fi

# per cada paràmetre, mostram el missatge Hola [paràmetre]
# amb for var in $* feim un recorregut per tots els paràmetres
for nom in $*
do
        echo "Hola $nom"
done
```

Exercici 7

Escriviu un shellscript que, a partir d'un únic paràmetre N, i utilitzant el bucle while mostra per pantalla una progressió aritmètica d'N termes (1, 2, 3 i 4...) i una progressió geomètrica d'N termes (1, 2, 4, 8, 16,...). El nombre de termes de les successions serà el $1 d'aquest shellscript.

```
#!/bin/bash

# Comprovam que s'hagi introduit un paràmetre numèric
if `test $# -ne 1`
then
        echo "ERROR: has d'introduir un paràmetre numèric."
        exit 1
fi

echo "Progressió aritmètica"

i=1 # contador. L'utilitzarem en el bucle, a cada interació s incrementarà en 1
num=1 # num és el resultat que mostrarem per pantalla

while `test $i -le $1`
do
        echo "$num"
        num=`expr $num + 1`
        i=`expr $i + 1`
done

echo "Progressió geomètrica"

i=1 # contador
num=1 # num és el resultat que mostrarem per pantalla
while `test $i -le $1`
do
        echo "$num"
        num=`expr $num \* 2 `
        i=`expr $i + 1`
done
```

Exercici 8

Escriviu un shellscript que, utilitzant el bucle while, mostri el factorial d'un nombre per pantalla. A continuació, feu el mateix amb un bucle until. El nombre sobre el qual calcular el factorial serà l'únic paràmetre d'aquest shellscript ($1).

Recordatori: el factorial d'un nombre és el producte de tots els nombre naturals des de 1 fins a aquest nombre. Per exemple, el factorial de 6 (6!) és 6·5·4·3·2·1.

```
#!/bin/bash

# comprovam que només s'hagi introduït un paràmetre
if `test $# -ne 1`
then
        echo "ERROR: introdueix un paràmetre numèric"
        exit 1
fi

i=$1 # guardam el factorial a una variable, a cada iteració del bucle, la decrementarem (-1)
fact=1
while `test $i -gt 1`
do
        fact=`expr $fact \* $i` # multiplicam el resultat per el nombre que es va decrementant
        i=`expr $i - 1`
done

echo "El factorial de $1 és $fact"
```

Exercici 9

Escriviu un shellscript que s'intenti autodestruir (com a fitxer). Abans d'executar aquest shellscript, feu una còpia de seguretat de la vostra feina, per si de cas. No podeu utilitzar el nom del fitxer per a eliminar-lo. Comenteu el comportament del shellscript i si s'esborra o no el fitxer.

```
#!/bin/bash

rm $0 # $0 conté el nom del fitxer

# si l'executam veim que s esborra el fitxer
```

Exercici 10

Escriviu un shellscript que, donats dos nombres diferents passats com a paràmetres mostri per pantalla els nombres que van des d'un fins l'altre (ambdós inclosos, creixentment o decrexentment). Cal que resoleu l'exercici amb un bucle until. Per exemple:

```
./exercici 5 20
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

```
#!/bin/bash

# comprovam que el nombre de paràmetres és 2
if `test $# -ne 2`
then
        echo "ERROR: introdueix dos paràmetres numèrics"
        exit 1
fi

# si el primer paràmetre és el menor que el segon, anirem sumant d'un a un, fins arribar al paràmetre 2
if `test $1 -lt $2`
then
        i=$1

        while `test $i -le $2`
        do
                printf "$i "

                i=`expr $i + 1`
        done
# si el segon es menor, anirem restant d'un a un fins arribar al paràmetre 2
else
        i=$1

        until `test $i -lt $2`
        do
                printf "$i "

                i=`expr $i - 1`
        done
fi

printf "\n"
```

Exercici 11

Escriviu un shellscript que, a partir del fitxer /etc/passwd, digui quin és l'intèrpret de comandes més utilitzat pels usuaris del sistema (és a dir, aquell que utilitzen més usuaris). Aquest shellscript s'ha de resoldre amb una única (i llarga) comanda i sense fer servir cap tipus de bucle.

```
#!/bin/bash

# Si s'introdueix algun paràmetre, tornam un error
if `test $# -ne 0`
then
        echo "Error: No s'ha d'introduir cap paràmetre"
        exit 1;
fi

# obtenim el terminal més utilitzat
cut -d: -f7 /etc/passwd | sort | uniq -c | sort -n -k1,1 | tail -n 1 | tr -s  " " | cut -d " " -f3
```

exit 0

Exercici 12

Elaboreu un shellscript que rebi com a paràmetres un nombre indeterminat de paraules en minúscula. El que haurà de fer el shellscript serà afegir cadascuna de les paraules a fitxers que s'anomenaran com la seva inicial.
Per exemple, si l'executem amb els paràmetres següents:

./exercici ratolí perifèric teclat pantalla

Al fitxer "r" s'afegirà la paraula "ratolí", al fitxer "p" s'afegirà la paraula "perifèric", al "t" teclat i al "p", novament, "pantalla".

```
#!/bin/bash

# Tornam un error i sortim si no s'ha passat cap paràmetre
if `test $# -eq 0`
then
        echo "ERROR: Introdueix almenys un paràmetre numèric"
        exit 1
fi

# feim un recorregut per tots els paràmetres
for p in $*
do
        # canviam les majúscules a minúscules
        p=`echo $p | tr '[:upper:]' '[:lower:]'`

        # obtenim la inicial
        inicial=$(echo $p | cut -c1)

        # guardam el paràmetre al fitxer corresponent
        # si no existeix el crearà
        # si ja està creat, guardarà el paràmetre
        # utilitzam >> per no trepitjar el contingut
        echo $p >> $inicial
done
```

Exercici 13

Indiqueu justificadament quina és la funció del següent shellscript, indicant quin és el significat més lògic dels paràmetres.

```
#!/bin/bash

for i in `sort $1`
do
   if grep $2 $i > /dev/null
   then
      echo A
      cp $i /tmp
      exit
   fi
done

echo B
```

```
#!/bin/bash

# fa un recorregut a la llista ordenada passada com a paràmetre
for i in `sort $1`
do
        # si existeix alguna línia a $i que contengui el paràmetre 2,
        # mostram A
        # copiam el contingut del fitxer $i al fitxer /tmp i  finalitza l'execució
        if grep $2 $i > /dev/null
        then
                echo A
                cp $i /tmp
                exit
        fi
done

# si arribam en aquest punt, vol dir que no s'ha trobat cap cadena de caràcters $2 dins cap fitxer de la llista passada per paràmetres
echo B

# veim que $1 és el nom d'un fitxer ordinari que conté una llista de noms de fitxers ordinaris
# $2 és una cadena de caràcters
```

Exercici 14

Escriviu un shellscript que, a partir de la ruta d'un directori determinat passada per paràmetre, si existeix, accedeixi a aquest directori.

- Si no existeix, haurà de crear aquest directori i accedir-hi.

- En cas de no poder crear el directori, mostrarà per pantalla el missatge "No s'ha pogut crear el directori" i quedar-se a la ruta original.

Resoleu aquest exercici sense utilitzar els operadors && i ||, gestionant la casuística amb estructures alternatives.

Per a comprovar que el vostre shellscript es comporta de la forma esperada, mostreu per pantalla el directori de treball actual. On queda la vostra terminal? On s'havia fet l'execució?

```
#!/bin/bash

# comprovam que s'hagi introduit un paràmetre, sino és així donam un missatge d error
if `test $# -ne 1`
then
        echo "ERROR: Has d'introduir la ruta d'un directori"
        exit 1
fi

# comprovam si el directori existeix
if `test -d $1`
then
        echo "El directori ja existeix"
        cd $1

# si no existeix el cream
# la següent línia intenta crear el fitxer i comprova si s'ha pogut crear a la vegada
elif `mkdir $1 2> /dev/null`
then
        echo "El directori s'ha creat"
        cd $1
else
        echo "No s'ha pogut crear el directori"
fi

pwd # mostram a quin directori ens trobam. En sortir de l'execució veurem que enstrobarem al mateix directori on esteim antes de l execució

exit 0
```

Exercici 15

Escriviu un shellscript que comprovi amb quin usuari s'ha executat. Si aquest s'ha executat com a superusuari, mostrarà el missatge “Sóc l'amo del món”. Si no s'ha executat inicialment com a superusuari, es tornarà a executar a si mateix com a superusuari.

En cap punt del shellscript pot aparèixer el nom del propi shellscript, caldrà que utilitzeu algun altre tipus de mecanisme o paràmetre per a poder executar-lo com a superusuari. Tingueu en compte que si s'ha executat algun sudo des del terminal recentment el resultat pot resultar confús: tanqueu i obriu el terminal per a assegurar-vos del resultat.

```
#!/bin/bash

# comprova que no s'hagi afegit cap paràmetre
if `test $# -ne 0`
then
        echo "ERROR: No s ha d'introduir cap paràmetre"
        exit 1
fi

# obtenim l'usuari
usuari=`ps -ef h | awk '{print $1,$2}' | grep $$ | cut -d " " -f1`
# amb ps -ef mostram tots els processos del sistema
# amb awk obtenim la primera i segona columna nom de l'usuari i identificador. La tercera és el procés pare i no el tenim en compte)
# amb grep filtram segons l'identificador del procés pid
# amb cut obtenim el camp 1

# comprovam si l'usuari és el root
# si no ho és, s'executa el propi shellscript però amb sudo.
# feim echo 'sudo... perquè el resultat del sudo serà mostrar un missatge i amb el echo el mostrarem a la terminal
if [ $usuari == "root" ]
then
        echo "Soc l'amo del mon"
else
        echo "Ho execut amb sudo"
        echo `sudo ./$0`
fi
```

Exercici 16

Escriviu un shellscript que es mati a si mateix (com a procés). Per a comprovar que funciona correctament, poseu alguna instrucció sota de l'acabament del procés i vegeu que no s'executa.

```
#!/bin/bash

# matam el procés
kill -9 $$

echo "Això no és veurà"

exit 0
```

Exercici 17

Escriviu un shellscript que a partir d'un nom de fitxer ($1) i un número de parts ($2), trenqui el fitxer $1 en $2 parts. Per a fer-ho caldrà que utilitzeu la comanda que trenca fitxers i alguna operació matemàtica. Proveu a trencar els fitxers en n parts i controleu que sempre es generi el nombre de fitxers que s'ha indicat, exactament.

```
#!/bin/bash

# comprovam si el nombre de paràmetres és 2
if `test $# -ne 2`
then
        echo "ERROR: El nombre de paràmetres és incorrecte"
        exit 1
fi

# per trencar el fitxer en $2 parts, hem de obtenir el tamany en bytes que hauria de tenir cada part
mida_total=`cat $1 | wc -c`
mida_fitxer=`expr $mida_total / $2`

# si feim l'operació d aquesta manera el residu de la divisió, per tant per fer el split ho hem de tenir en compte
residu=`expr $mida_total % $2`

# d'aquesta manera no deixarem cap byte a mitjes
if `test $residu -gt 0`
then
        mida_fitxer=`expr $mida_fitxer + 1`
fi

# trencam el fitxer en fitxers de $mida_fitxer bytes
split -b $mida_fitxer $1 part_

exit 0;
```

Exercici 18

Escriviu un shellscript que, donat un fitxer (amb la seva ruta absoluta) i el nom d'un paquet, ens indiqui si el fitxer pertany al paquet o no (és a dir, si va ser instal·lat o requerit pel paquet).

```
#!/bin/bash

# Comprobam si el nombre de paràmetres és 2
if `test $# -ne 2`
then
        echo "ERROR: El nombre de paràmetres és incorrecte"
        exit 1
fi

# per comprovar si pertany o no,
# obtenim la llista de fitxers del paquet $1
# filtram per trobar si un dels fitxers s'anomena $2
# amb &> redireccionam la sortida estàndard i la d'error a /dev/null perquè no es vegi al terminal
if dpkg -L $1 | grep ^$2$ &> /dev/null
then
        echo "$2 pertany al paquet $2"
else
        echo "$2 no pertany al paquet $2"
fi

exit 0
```

Exercici 19

Teniu un fitxer de text on, a cada línia, hi ha dos noms de fitxer (separats per guió "-"). Escriviu un shellscript que processi totes les línies d'aquest fitxer efectuant la següent operació: si el primer fitxer existeix, el copiarà sobre el segon; si no existeix, copiarà el contingut del fitxer /etc/group sobre el segon.

```
#!/bin/bash

# Comprobam si el nombre de paràmetres és 1
if `test $# -ne 1`
then
        echo "ERROR: El nombre de paràmetres és incorrecte"
        exit 1
fi

# feim un recorregut per les línies del fitxer $1
for l in `cat $1`
do
        # obtenim el fitxer 1 i el fitxer 2 indicat a la línia
        fitxer1=`echo $l | cut -d"-" -f1`
        fitxer2=`echo $l | cut -d"-" -f2`

        # si el fitxer 1 existeix, es copia el seu contingut sobre el segon
        if `test -f $fitxer1`
        then
                cp $fitxer1 $fitxer2
        # si no existeix, es copia el contingut de /etc/group sobre el fitxer 2
        else
                cp /etc/group $fitxer2
        fi
done

exit 0
```

Exercici 20

Indiqueu justificadament quina és la funció del següent shellscript, indicant quin és el significat més lògic dels seus paràmetres.

```
#!/bin/bash

j=0

for i in $*
do
   if grep ttt $i > /dev/null 2> /dev/null
   then
      j=`expr $j + 1`
   fi
done

echo $# $j
```

Digues quin efecte tindria substituir la línia en blau per: if grep ttt $i

```
#!/bin/bash

# assignam el valor 0 a j
j=0

# feim un recorregut pels paràmetres
for i in $*
do

        # comprovam que $i tengui la cadena "ttt"
        # amb > /dev/null indicam que no es vegi la sortida estàndand
        # amb 2> /dev/null indicam que no es vegi la sortida d'errors
        # aquests 2 redireccionals són equivalents a fer &> /dev/null
        if grep ttt $i > /dev/null 2> /dev/null
        then
                # sumam 1 a j
                # en sortir del bucle, j tindrà el nombre d'arxius que contenten al menys una línia amb la cadena de caràcters ttt
                j=`expr $j + 1`
        fi

done

echo $# $j

# Els paràmetres són noms de fitxers i feim el grep per comprovar si alguns tenen a una de les seves línies ttt

# Si eliminam els redireccionament, apareix el resultat de grep al terminal
```

Exercici 21

Escriviu un shellscript que a partir d'un directori i una paraula, passats com a paràmetres:

a) comprova que el primer paràmetre sigui un directori:
b) En cas que no ho sigui, mostrarà un missatge d'error.
c) En cas que ho sigui cercarà a tots els fitxers del directori indicat (només el directori, no els seus subdirectoris) la paraula passada com a paràmetre i indicarà, per a cada fitxer, el nombre d'ocurrències de la paraula que s'ha indicat.

Resoleu aquest exercici utilitzant un bucle **for**. No és permet l'ús de la comanda **find**.

```
#!/bin/bash

# comprovam que el nombre de paràmetres sigui 2
if `test $# -ne 2`
then
        echo "ERROR: El nombre de paràmetres no és correcte"
        exit 1
fi

# comprovam que $1 sigui un directori
if `test -d $1`
then
        # recorrem tots els fitxers del directori $1
        for f in $1/*
        do
                # mostram el nom del fitxer i el nombre de ocurrències de $2
                # amb -o tindrem en compte el nombre de vegades que apareix la paraula, si utilitzam només comptarà una paraula per línia màxim
                # -i no es és case sentitive
                echo -n "$f "
                grep -o -i $2 $f | wc -l
        done
else
        echo "ERROR: El primer paràmetre ha de ser un directori"
        exit 1
fi

exit 0
```

Exercici 22

Escriviu un shellscript que rebi com a paràmetre el nom d'un directori i doni com a resultat un únic fitxer, a partir de la compressió i empaquetament del directori i tots els seus subdirectoris.

a) Feu les comprovacions necessàries per tal que el shellscript comprovi si el directori a comprimir existeix.
b) Feu les comprovacions necessàries per tal que només admeti directoris (no fitxers, per tant).
c) Feu que el nom del fitxer acabi amb la data actual, en el format AAAA_MM_DD. Per exemple, el backup del 10 de maig de 2021, seria backup_2021_05_10.tar.gz. Caldrà que s'emmagatzemi en la ruta que crida el shellscript.

Si a l'hora de comprimir se us mostren missatges d'error (típicament referents a la falta de permisos), caldrà que silencieu la sortida.

```
#!/bin/bash

# comprovam que el nombre de paràmetres sigui 1
if `test $# -ne 1`
then
        echo "ERROR: El nombre de paràmetres no és correcte"
        exit 1
fi

# comprovam si existeix el directori
if `test -d $1`
then
        # obtenim el nom i el comprimim
        nom=`date +"backup_%Y_%m_%d.tar.gz"`
        tar -cvzf $nom $1 2> /dev/null
else
        echo "ERROR: El firectori indicat no existeix"
        exit 1
fi
```

Exercici 23

Escriviu un shellscript que rebi com a paràmetre un valor alfanumèric V. Aquest shellscript haurà de posar el sistema en el **runlevel** indicat. Abans de posar-lo en aquest runlevel, caldrà que comprovi que, efectivament, aquest és un runlevel legal. En cas que aquest runlevel no existeixi, el sistema mostrarà un missatge per pantalla indicant que no s'ha pogut executar l'acció. En cas que existeixi, el sistema canviarà el runlevel a l'indicat.

Tingueu en compte el tipus d'usuari per a executar aquest shellscript i les comprovacions necessàries per a poder executar-lo.

```
#!/bin/bash

# comprovam que el nombre de paràmetres sigui 1
if `test $# -ne 1`
then
        echo "ERROR: El nombre de paràmetres no és correcte"
        exit 1
fi

# obtenim el nom d'usuari
usuari=`whoami`

# comprovam si el runlevel passat per paràmetre és o no correcte
case $1 in
        0|1|s|S|2|3|4|5|6)
                # si es correcte, si no som l'usuari root executarem init amb sudo
                if [[ $usuari == 'root' ]]
                then
                        init $1
                else
                        sudo init $1
                fi
                ;;
        *)
                echo "ERROR: El runlevel indicat no és correcte"
                exit 1
                ;;
esac

exit 0;
```

Exercici 24

Indiqueu justificadament quina és la funció del següent shellscript, indicant quin és el significat més lògic dels paràmetres.

```
#!/bin/bash

for j in ../directori/*
do
    if tail $j | grep $1 > /dev/null
    then
        mv $j .
    fi
done
```

```
#!/bin/bash

# recorrem tots els fitxers dins el directori ../directori
# recordam que .. significa el directori pare del directori actual
# per tant, ../directori es troba dins el directori pare
for j in ../directori/*
do
        # si a les darreres 10 línies del fitxer $j, es troba al menys una vegada la cadena de caràcters $1, es mou el fitxer al directori actual
        if tail $j | grep $1 > /dev/null
        then
                mv $j .
        fi
done

# $1 és una cadena de caràcters
```

Exercici 25

Escriviu un shellscript que a partir d'una ruta de directori (absoluta o relativa) passada per paràmetre, examini tots els fitxers d'extensió *.c del directori indicat i compti el nombre de línies no blanques. La sortida del shellscript haurà de ser un fitxer "linies.txt" que contindrà els noms de cada fitxer, junt al nombre de línies no buidas d'aquest, ordenat de forma decreixent segons el nombre de línies buides.

```
$ ./exercici ../practiquesC
$ cat linies.txt
practica3.c 1530
```

```
# comprovam que el nombre de paràmetres sigui 1
if `test $# -ne 1`
then
        echo "ERROR: El nombre de paràmetres no és correcte"
        exit 1
fi

# recorrem els fitxers .c de la ruta indicada per paràmetrem
for f in $1/*.c
do
        # obtenim el nombre de línies no buides i afegim el resultat al fitxer linies.txt
        linies=`cat $f | grep -c .`
        echo  "$f $linies" >> linies.txt
done

# ordenam numèricament segons el nombre de línies. Comprimim els / repetits consecutivament
cat linies.txt | sort -n -r -k2 | tr -s "/" > linies.txt
```

Exercici 26

Escriviu un shellscript que, per a totes les paraules d'un fitxer hipotètic, paraules.txt, cerqui si hi ha alguna de les paraules del fitxer dins els fitxers del vostre directori actual (i el seu arbre de directoris complets) que contingui qualsevol de les paraules del fitxer indicat. En concret, caldrà que, per a cada paraula, digueu quants fitxers la contenen. Podeu utilitzar la comanda **find** per fer la cerca.

$ ./exercici paraules.txt
$ arbre 7
$ casa 10
$ cotxe 4

```
#!/bin/bash

# comprovam que no s'hagi passat cap paràmetre
if `test $# -ne 0`
then
       echo "ERROR: No s'ha d'introduir cap paràmetre"
       exit 1
fi

# feim un recorregut per les línies del fit
for p in `cat paraules.txt`
do
       echo -n "$p "
       # indicam que ha de trobar al directori actual (.) els fitxers de tipus regular (-type f), executant la comanda grep per filtrar per la paraula
       # quan obtenim les línies, les contam utilitzant wc -l
       find . -type f -exec grep -l "$p" {} \; | wc -l
done

exit 0
```

Exercici 27

Indiqueu justificadament quina és la funció del següent shellscript, indicant el significat més probable dels seus arguments.

```
#!/bin/bash

if test -f $1
then
    if test `wc -c < $1` -gt $2
    then
        gzip $1
    fi
fi
```

```
#!/bin/bash

#!/bin/bash

# comprova si $1 és un fitxer
if test -f $1
then
        # si ho és, es comprova que la mida del fitxer $1 és més gran que el paràmetre $2
        # si és així, comprimeix el fitxer
        if test `wc -c < $1` -gt $2
        then
                gzip $1
        fi
fi
```

Exercici 28

Escriviu un shellscript que, donat un nom d'usuari mostri per pantalla a quins grups pertany, ja sigui de forma primària o secundària, és a dir, escriviu una comanda que faci el mateix que la comanda groups, per a qualsevol usuari. Heu de resoldre l'exercici manipulant els fitxers /etc/passwd i /etc/group i no podeu utilitzar les comandes group i id.

```
#!/bin/bash

# comprovam que no el nombre de paràmetres sigui 1
if `test $# -ne 1`
then
        echo "ERROR: El nombre de paràmetres introduits és incorrecte"
        exit 1
fi

# mostram el grup primari
GID=`cat /etc/passwd | grep ^$1 | cut -d: -f4`

# comprovam si l'usuari existeix
if `test $GID -ge 0 2> /dev/null`
then
        echo -e "\nGID grup primari: $GID"

        # obtenim el nom del grup
        GID_nom=`cat /etc/group | grep [:..]$GID[:..] | cut -d: -f1`

        echo "Nom grup primari: $GID_nom"

        # mostram els grups secundaris
        echo -e "\nGrups secundaris: "
        gsec=`cat /etc/group | grep [:..]"\<$1\>" | cut -d: -f1`

        # avisam a l usuari si no s han trobat grups secundaris, si s'han trobat, els mostram
        numgsec=`echo -e "$gsec" | wc -l`

        if `test $numgsec -eq 0`
        then
                echo "L'usuari $1 no té grups secundaris"
        else
                echo -e "$gsec"
        fi
fi
```

Exercici 29

Escriviu un shellscript que mostri els noms dels fitxers ordinaris (no altres directoris) del directori actual amb mida igual o superior a la mida especificada per l'usuari com a paràmetre.

Recomanació: escriviu un bucle que iteri mostrant els noms de tots els fitxers del directori actual. A continuació, feu que a cada iteració es prengui la mida del fitxer. Finalment, si aquesta mida és superior al paràmetre del shellscript, mostreu per pantalla el nom del fitxer.

```
#!/bin/bash

# comprovam que no el nombre de paràmetres sigui 1
if `test $# -ne 1`
then
        echo "ERROR: El nombre de paràmetres introduits és incorrecte"
        exit 1
fi

# recorrem els fitxers del directori actual
for f in *
do
        # si el fitxer és un fitxer regular, obtenim la seva mida i comprovam si és igual o major que $1
        # si és aixi, mostram el nom del fitxer
        if `test -f $f`
        then
                mida=`wc -c $f | cut -d" " -f1`
                if `test $mida -ge $1`
                then
                        echo $f
                fi
        fi
done
```

Exercici 30

Escriviu un shellscript que faci una còpia al directori updated tots els fitxers ordinaris del directori actual i els seus subdirectoris que hagin estat modificats al llarg de les darreres 48 hores (per fer proves pot resultar-vos útil la comanda touch que permet modificar les dades associades a un fitxer). Assegureu-vos que el directori updated existeix i, si no, creeu-lo.

```
#!/bin/bash

# Comprovació del nombre de paràmetres
if `test $# -ne 0`
then
        echo "ERROR: Nombre de paràmetres incorrecte"
        exit 1
fi

# si no existeix el directori, el cream
# amb el caràcter !, ens ficam dins l'if si la condició és falsa
if `test ! -d updated`
then
        mkdir updated;
fi

# recorrem tots els fitxers regulars del directori actual que s'han modificat en les darreres 48 hores
# -mtime -2 (24*N)
# i els copiam al directori updated
for fitx in `find . -type f -mtime -2`
do
        cp $fitx updated/
done

exit 0
```

Unitat 5: repàs per l'examen

1. Escriviu un shellscript que, a partir d'una ruta de directori indicada dins un fitxer (el nom del fitxer es passa per paràmetre $1) examini si existeix un fitxer anomenat el nom indicat pel paràmetre $2 a la ruta indicada pel fitxer $1. Si és així, ha de mostrar un missatge que indiqui que existeix. Si no és així, l'ha de crear a la ruta $1. Si no és possible crear el fitxer, ha de mostrar un missatge que indiqui que no s'ha pogut crear el fitxer.

NOTA: el fitxer passat per paràmetre només conté la ruta del directori.

```
#!/bin/bash

# si no s'han passat 2 paràmetres, error
if `test $# -ne 2`
then
        echo "ERROR: s'han de passar dos paràmetres"
        exit 1
fi

# comprovam que $1 és un fitxer. Si és així, obtenim la ruta del directori.
# aquesta comprovació no és obligatòria
if `test -f $1`
then
        ruta=`cat $1`
else
        echo "ERROR: $1 no és un fitxer regular o no existeix"
        exit 1
fi

cd $ruta

# comprovam si existeix
if `test -f $2`
then
        echo "El fitxer $1 ja existeix dins $ruta"
# intentam crear el fitxer
elif `touch $2 2> /dev/null`
then
        echo "El fitxer $2 s'ha creat dins $ruta"
else
        echo "El fitxer $2 no s'ha pogut crear dins $ruta"
fi

exit 0
```

2. Escriviu un shellscript que:
   - Demana a l'usuari que indiqui una operació que pot ser: + - / % * q. L'usuari ha d'introduir el caràcter corresponent per teclat.
   - Si posa **q**, l'usuari està indicant que vol sortir. Per tant, el shellscript s'ha de despedir i acabar.
   - Si no ha posat cap d'aquestes opcions, s'ha d'indicar que no és correcte i tornar a demanar a l'usuari que indiqui una operació.
   - Si posa + - / % *, s'ha de realitzar l'operació corresponent amb els paràmetres $1 i $2, i donar un missatge amb la solució a l'usuari.
   - Després d'això tornarà a demanar que a demanar que l'usuari afegesqui una opció.

Obligatori afegir comentaris i comprovació de paràmetres.

```
#!/bin/bash

# comprovació de paràmetres
if `test $# -ne 2`
then
        echo "ERROR: nombre de paràmetres incorrecte"
        exit 1;
fi

# posam un bucle infinit per demanar sempre a l'usuari que introueixi una opció. Sortirem amb l opció q
while `true`
do
        read -p "Introdueix una de les següents opcions + - / % * q: " opcio

        case $opcio in
        "+")
                echo "El resultat és "`expr $1 + $2`
                ;;
        "-")
                echo "El resultat és "`expr $1 - $2`
                ;;
        "/")
                echo "El resultat és "`expr $1 / $2`
                ;;
        "%")
                echo "El resultat és "`expr $1 % $2`
                ;;
        "*")
                echo "El resultat és "`expr $1 \* $2`
                ;;
        "q")
                echo "Hasta luego!"
                exit 0
                ;;
        *)
                echo "Cap de les opcions és correcta"
                ;;
        esac
done
```