

# SCRIPTS SHELL

## INTRODUCCIÓ

Un **script shell** (en anglès shell script) son un **seguit de comandes** i crides que l'interpret de comandes del sistema operatiu s'encarrega d'executar seqüencialment. D'aquesta manera es poden construir programes emprant comandes Linux/Unix, executables propis, etc, utilitzant un o diversos fitxers.

En aquest temari, es farà servir un llenguatge interpretat anomenat **bash**. Vol dir que s'executa línia a línia i si hi ha un error, es parará en aquella línia. També es case sensitive i no fa falta ;

## CREACIÓ D'UN SCRIPT

```
server@server:~$ vim primerScript.sh
```

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

```
#Per a crear un script, s'ha d'indicar amb  
#la comanda que apareix a l'esquina superior
```

```
echo "Hola, món" _
```

## PERMISOS D'EXECUCIÓ

Per donar permisos d'execució, utilitzarem la comanda **chmod**.

**chmod -u/g/o- -+/- -r/w/x- -fitxer-** afegeix o treu permisos a **(u)ser/(g)roup/(o)ther** dependent (+ o -) dels permisos **(r)ead/(w)rite/(e)ecute** del fitxer.

```
server@server:~$ chmod u+x primerScript.sh
```

## EXECUCIÓ D'UN SCRIPT

```
server@server:~$ ./primerScript.sh
Hola, món
```

Per a executar el nostre script, és indispensable començar la comanda amb `./`.  
Tot i que no sigui obligatori, és recomanable posar l'extensió `.sh` als scripts bash.

## VARIABLES EN BASH

Una variable és **espai de memòria** on s'emmagatzema **informació**.  
Les variables en bash no tenen tipatge, s'assignen directament la informació.

La declaració de variables no ha de tenir espais.

```
server@server:~$ nom=Pau
```

Per consultar el valor de la variable emprarem el símbol `$` davant del nom de la variable.

```
server@server:~$ nom=Pau
server@server:~$ echo $nom
Pau
```

```
#!/bin/bash

x="Hola, món"
echo $x_

v=1337
z=$v
echo $z
```

Podem assignar una **cadena de caràcters** a una variable, tal i com mostra l'exemple i, també podem assignar el contingut d'una **variable** a una **altra variable**.

També, podem **assignar** el resultat de **comandes** dins de **variables**.

Per a fer-ho, haurem de seguir l'estructura següent: **var=\$(comanda)**.

```
server@server:~$ variable=$(ls)
server@server:~$ echo $variable
AltreExemple.txt exempleScript.sh hola.txt primerScript.sh
```

## DIFERÈNCIA ENTRE “ ” | ‘ ’

El primer i el segon **echo** imprimiran **33**, mentres que el 3r **echo** imprimirà **\$nombre**.

```
#!/bin/bash
nombre=33
echo $nombre
echo "$nombre"
echo '$nombre'

server@server:~$ vim exempleScript.sh
server@server:~$ chmod u+x exempleScript.sh
server@server:~$ ./exempleScript.sh
33
33
$nombre
server@server:~$

#!/bin/bash

var=33

echo "Per verure el contingut de la variable, hem d'escriure un $ d
avant de la variable: echo \$var. Exemple: $var"
```

## PARÀMETRES EN BASH

Existeixen una sèrie de variables especials. Aquestes, permeten obtenir informació útil de cara a controlar diferents tasques dins de l'script.

```
#!/bin/bash

#Script que ens mostra parametres posicionals

echo $0 #mostra el nom del fitxer
echo $1 #mostra el primer parametre
echo $2 #mostra el segon parametre
echo $3 #mostra el tercer parametre

echo $# #mostra el numero de parametres que se li passa al script
echo $* #mostra tots els parametres que se li passa al script
echo $$ #mostra el id del process (pid) associat a l'execucio
```

Al executar el nostre script, passant uns paràmetres, ens donaria el següent resultat.

```
pau@pauserver:~$ ./posicionals.sh pep 34 barcelona
./posicionals.sh
pep
34
barcelona
3
pep 34 barcelona
1142
```

# OPERACIONS EN BASH

**var=\$((operació aritmètica))** → Assigna a la variable **var** el resultat de l'operació.

En aquest exemple, haurem d'executar el script amb dos paràmetres, que seran els dos nombres a operar. → **./operacioAritmetica.sh 4 2**

```
#!/bin/bash

suma=$(( $1 + $2 ))

echo "La suma de $1 i $2 es $suma"
```

Al executar el script, sortiria el següent: **La suma de 4 i 2 es 6.**

Per a mostrar decimals, farem servir la comanda

**echo -e "scale=nDecim;valor1/valor2" | bc -l.** Molt útil a l'hora de dividir, ja que s'arrodoneix.

```
pau@pauserver:~/scripts$ echo -e "scale=3;4/3" | bc -l
1.333
```

# LECTURA PER TECLAT

Per llegir una variable, es posa **read** i el nom de la variable per guardar-la. Més endavant, si volem executar aquesta variable, haurem de fer servir l'estructura tradicional **\$variable**.

```
#!/bin/bash

echo -n "Primer valor a sumar: "
read num1

echo -n "Segon valor a sumar: "
read num2

suma=$(echo "scale=2;$num1+$num2" | bc -l)

echo "$num1 + $num2 = $suma"
```

## ECHO

Les cadenes de caràcters en bash poden estar, com hem vist anteriorment, tant entre cometes dobles com entre cometes simples (la diferència, recordem, està en que les primeres interpreten les variables i les segones no).

Podem inserir caràcters especials com tabulacions amb `\t` o salts de línia amb `\n`. Però, per tal d'indicar que volem que s'interpretin aquests caràcters especials haurem de passar-li l'opció **-e** a la comanda echo.

```
#!/bin/bash

echo -e "Abans de tabular \t Despres de tabular"
echo -e "Farem salts de linia \n un salt\n dos salts"
```

# ESTRUCTURES DE CONTROL

## CONDICIONALS

### IF

L'estructura de control IF en bash té la següent estructura.

```
#!/bin/bash

echo -n "Introdueix el primer numero: "
read num1

echo -n "introdueix el segon numero: "
read num2

if [ $num1 -gt $num2 ]
then
    echo "El $num1 es mes gran que el $num2"
elif [ $num1 -eq $num2 ]
then
    echo "El $num1 i $num2 son iguals"
else
    echo "El $num1 es mes petit que el $num2"
fi
```

**Indispensable posar espais** entre els claudàtors i la informació a processar.

Abans de començar a escriure la condició, s'ha d'escriure ***then***.

Per a tancar el condicional, s'ha d'escriure ***fi***.

Fa servir els següents comparadors numèrics.

A > B	-gt
A < B	-lt
A >= B	-ge
A <= B	-le
A = B	-eq
A != B	-ne

O comparadors de strings:

A = B	A == B
A != B	A != B
String vació	-z
String no vació	-n

## CASE

**Exercici:** Crea un script que interpreti quin input ha introduït l'usuari.

```
#!/bin/bash

echo -n "Introdueix un numero: "
read valor

case $valor in
    1|5)
        echo "Has pulsat el 1 o el 5"
        ;;
    2)
        echo "has pulsat el 2"
        ;;
    3)
        echo "has pulsat el 3"
        ;;
    [6-8])
        echo "Has pulsat el 6, 7 o 8"
        ;;
    *)
        echo "has pulsat un numero diferent del 1,2,3,5,6,7,8"
        ;;
esac
```

# LOOPS

## WHILE

**Exercici:** Crea un script on l'usuari hagi d'introduir un nombre fins que l'encerti.

```
#!/bin/bash

numEncertar=24
contador=0
num1=-1

while [ $num1 -ne $numEncertar ]
do
    echo -n "Introdueix un numero entre el 1 i 99: "
    read num1
    clear
    contador=$((contador+1))
    if [ $num1 -gt $numEncertar ]
    then
        echo "$num1 es mes gran que el numero secret"
    elif [ $num1 -lt $numEncertar ]
    then
        echo "$num1 es mes petit que el numero secret"
    else
        echo "Has encertat el numero secret!"
        echo "L'has encertat en $contador intents"
    fi
done
```

També podem llegir les **línies d'un fitxer** amb la següent estructura.

```
#!/bin/bash

while read linea
do
    echo $linea
done < lista.txt
```

## RANDOM

**\$RANDOM** → Dóna un valor entre el 0 i 32767

Per fer-ho entre dos números és  $\$((($RANDOM + <num1>)%<num2>))$  -> random entre num1 i num2 -1.

```
numEncertar=$(( ($RANDOM + 1) % 100 ))
```



## FOR

**Exercici:** Contar el nombre de paràmetres introduïts.

```
#!/bin/bash

contador=0

for par in $*
do
    contador=$((contador+1))
    echo -e "Parametre $contador: $par"
done
```

**Exercici:** Amb la comanda **ls**, fes un script que digui si els fitxers són directori o no.

```
#!/bin/bash

for i in $(ls)
do
    if [ -d $i ]
    then
        echo "$i es un directori"
    else
        echo "$i es un fitxer"
    fi
done
```

## GREP

És una eina de línia d'ordre usada en sistemes Linux i Unix per **cercar una paraula** o **patró** específic en un **fitxer** o grup de fitxers. Podríem dir que funciona com un ctrl + F.

## PASSWORD

cat /etc/passwd -> fitxer important on diu quins usuaris hi ha en el sistema  
Per accedir a la documentació del fitxer passwd es: **man 5 passwd**.

pau@pauserver: ~

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
```

El **delimitador** del camp es el **:**.

El primer camp es el **nom del login**.

El segon camp es la **contrasenya**.

El tercer camp es el **ID de l'usuari**.

El quart camp es el **ID del grup**.

El cinquè camp es una **descripció de l'usuari**.

El sisè camp és el directori **home** de l'usuari.

El setè camp és l'**intèrpret shell** de l'usuari.

```
pau@pauserver:~$ cat /etc/passwd |grep bash
root:x:0:0:root:/root:/bin/bash
pau:x:1000:1000:pau:/home/pau:/bin/bash
pau@pauserver:~$
```

**grep -w:** li diu que només busqui aquella paraula.

```
pau@pauserver:~$ cat /etc/passwd |grep usuari
usuari1:x:1001:1001:usuari de proves 1,408,982783432,:/home/usuari1:/bin/bash
pau@pauserver:~$ cat /etc/passwd |grep -w usuari
usuari1:x:1001:1001:usuari de proves 1,408,982783432,:/home/usuari1:/bin/bash
```

## AWK

La comanda Awk ens permet **seleccionar** una **columna determinada** i mostrar-la a la pantalla per poder **visualitzar** la informació. Per a fer-ho, haurem de cridar el fitxer amb la comanda:

**–fitxer– | awk '{print \$–numero–}'**

Aquesta comanda permet treure totes les dades del nombre d'aquella columna.

Podem, per exemple, llegir les dades del següent document anomenat **dades.txt**.

```
server@server:~$ cat dades.txt

Jose Hernandez Sanz 34212312H si
Luis Garcia Alonso 12212323K no
Amadeu Holf Ramirez 43214321J si
Carlos Andres Lopez 11122233D no
```

Cridant al fitxer amb la comanda **cat**, insertem després de la **pipe** la comanda **awk**.

En aquest cas, li diem que imprimeixi la columna número 1, amb el paràmetre **print \$1**.

```
server@server:~$ cat dades.txt | awk '{print $1}'

Jose
Luis
Amadeu
Carlos
```

També, podem filtrar dades afegint la condició següent. En aquest exemple, posem la condició de que només mostri els noms –columna 1– que a la cinquena columna mostri “no”.

```
server@server:~$ cat dades.txt | awk '$5=="no" {print $1}'

Luis
Carlos
```

A part, podem personalitzar el delimitador de columnes, que per defecte és un espai “ ”.

Per canviar-ho, es fa amb la comanda: **awk -F–delimitador–**.

# EXERCICIS

## EXERCICI 1

Fes un script que ens indiqui si la tecla que hem polsat és una lletra, un número o un altre caràcter. Fes-ho amb la comanda “case”

Ex: un rang de valors del 0 al 9 es pot avaluar amb [0-9] .

```
do

    echo "Escriu un caracter"
    read char

    case $char in
        [0-9])
            echo "El caracter es un numero"
            ;;
        [A-z])
            echo "El caracter es una lletra"
            ;;
        *)
            echo "El caracter no es ni un numero ni una lletra"
            ;;
    esac
    echo "Vols repetir?(y/n)"
    read userRepeat
    case $userRepeat in
        'Y'|'y'|'Yes'|'YES'|'yes')
            repeat=1
            ;;
        *)
            repeat=0
            ;;
    esac
done
```

## EXERCICI 2

Fes un script que quan l'executem ens digui quin dia de la setmana és (dilluns, dimarts, dimecres, dijous, divendres, dissabte o diumenge). Utilitza la comanda “date” per saber el dia de la setmana.

```
dayCat=""

case $day in
    "Monday")
        dayCat="Dilluns"
        ;;
    "Tuesday")
        dayCat="Dimarts"
        ;;
    "Wednesday")
        dayCat="Dimecres"
        ;;
    "Thursday")
        dayCat="Dijous"
        ;;
    "Friday")
        dayCat="Divendres"
        ;;
    "Saturday")
        dayCat="Dissabte"
        ;;
    "Sunday")
        dayCat="Diumenge"
        ;;
esac

time=$(date | awk '{print $4}')
hours=$(echo $time | awk -F: '{print $1}')
minutes=$(echo $time | awk -F: '{print $2}')
seconds=$(echo $time | awk -F: '{print $3}')

echo "Avui estem a $dayCat i son les $hours hores $minutes minuts i $seconds segons"
```

### EXERCICI 3

Fes un script que ens doni la hora escrita segons el següent format: “dotze hores i vint-i-tres minuts”.

```
#!/bin/bash

time=$(date | awk '{print $4}')
hour=$(echo $time | awk -F: '{print $1}')
minute=$(echo $time | awk -F: '{print $2}')
second=$(echo $time | awk -F: '{print $3}')

writtenHour=$(awk -F: -v h=$hour ' $1==h {print $2}' numeros.txt)
writtenMinute=$(awk -F: -v m=$minute ' $1==m {print $2}' numeros.txt)
writtenSecond=$(awk -F: -v s=$second ' $1==s {print $2}' numeros.txt)

echo "Son les $writtenHour hores $writtenMinute minuts i $writtenSecond segons"
```

### EXERCICI 4

Fes un script on calculi el factorial del número introduït.

```
#!/bin/bash

echo "Introdueix un numero"
read num

value=1

for i in `seq $num -1 1`;
do
    value=$(( $value * i ))
done

echo "El factorial de $num es $value"
```

## EXERCICI 5

Fes un script on digui si el número introduït és un número primer o no

```
#!/bin/bash

echo "Introdueix un número per saber si és un número primer"
read num
prime=1

for i in $(seq 2 $((num - 1)))
do
    if [ $((num % i)) -eq 0 ]
    then
        prime=0
        i=$num
    fi
done

if [ $prime -eq 1 ]
then
    echo "El numero $num es primer"
else
    echo "El numero $num no es primer"
fi
```

## EXERCICI 6

Fes una funció per dir si un número és primer o no i comproba quins números del 2 al 1000 són primers

```
#!/bin/bash

esPrimer()
{
prime=1

for i in $(seq 2 $(( $1 - 1 )))
do
    if [ $(( $1 % $i )) -eq 0 ]
    then
        prime=0
        i=$num
    fi
done
return $prime
}

for num in $(seq 2 1000)
do
    esPrimer $num

    if [ $? -eq 1 ]
    then
        echo -n "$num, "
    fi
done

echo ""
```

Crear una funció: <nomFuncio>() { codi **return** valor }

Cridar funció: <nomFuncio> <valor (si admet valor)>

\$? evalua el valor que retorna la funció

## EXERCICI 7

Treu els IDs i noms dels usuaris que hi ha en la màquina.

```
#!/bin/bash

echo $(awk -F: '$3 >= 1000 && $3 < 65534 {print $1, $3}' /etc/passwd)
```

## EXERCICI 8

Desde un paràmetre posicional, dona la ruta d'un directori i de tots els fitxers d'aquell directori, mostra només els noms dels directoris

```
#!/bin/bash

if [ -d $1 ]
then
    for file in $(ls $1)
    do
        if [ -d "$1/$file" ]
        then
            echo "$file"
        fi
    done
else
    echo "$1 no es un directori"
fi
```



## EXERCICI 9

Fes un script que ens busqui un fitxer dins d'un directori . En cas que existeixi ens digui el propietari del mateix. El script ens demanarà un nou directori, en cas que el que fiquem no existeixi.

```
#!/bin/bash

boolean=0

while [ $boolean -ne 1 ]
do
    echo "introdueix un path amb un fitxer"
    read path
    if [ -e $path ]
    then
        boolean=1
        propietari=$(ls -l $path | awk '{print $3}')
        fitxer=$(ls -l $path | awk '{print $9}')
        echo "El propietari de $fitxer és $propietari"
    else
        echo "El directori no existeix, torna'l a introduir"
    fi
done
```

## TAULES DE MULTIPLICAR

Fer un script que ens doni com a resultat un fitxer de text en el que tindrem les taules de multiplicar del 0 al 10.

```
#!/bin/bash

echo "Taula del $1:" > taula_$1.txt
for i in $(seq 0 10)
do
    echo "$1 x $i = $(( $1 * $i ))" >> taula_$1.txt
done

~
```

```
#!/bin/bash


echo "Creació del document" > TaulesMultiplicar.txt

for i in $(seq 0 10)
do
    echo "S'ha creat la taula del $i a TaulesMultiplicar.txt"

    for numero in $(seq 0 10)
    do
        echo "$i * $numero = $((i*$numero))" >> TaulesMultiplicar.txt
    done
    echo "" >> TaulesMultiplicar.txt
done
```

## EXERCICI 10

Fes ping a varies direccions. Després, mostra els PID dels pings i fes que amb un número, puguis matar els processos.

 pau@pauserver: ~/scripts/exercicis

```
#!/bin/bash

numero=1

for i in $(pidof ping)
do
    array_ping[$numero]=$i
    echo -n "$numero: "
    ps -ef | awk '$2=="'$i'" {print $8, $9}'
    numero=$((numero+1))
done
echo -n "Quin ping vols aturar: "

read linia
kill -9 ${array_ping[$linia]}

~
```

# CREACIÓ D'UN USUARI

```
pau@pauserver:~$ sudo adduser usuari1
[sudo] password for pau:
info: Adding user `usuari1' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `usuari1' (1001) ...
info: Adding new user `usuari1' (1001) with group `usuari1 (1001)' ...
info: Creating home directory `/home/usuari1' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for usuari1
Enter the new value, or press ENTER for the default
  Full Name []: usuari de proves 1
  Room Number []: 408
  Work Phone []: 982783432
  Home Phone []:
  Other []:
Is the information correct? [Y/n] y
info: Adding new user `usuari1' to supplemental / extra groups `users' ...
info: Adding user `usuari1' to group `users' ...
```

## LINUX

Linux es multiusuari i permet multiprocés, és a dir, permet executar múltiples tasques a l'hora. Una d'aquestes tasques està en primer pla (activa) mentre que totes les altres estan en segon pla.

**Ctrl + Z:** adorm un procés.

**bg:** executa el procés en segon pla.

**fg:** posa el procés en primer pla.

**Ctrl + C:** para el procés en primer pla.

procés > /dev/null : executa el procés sense que es vegi per pantalla

procés > /dev/null & : executa el procés sense que es vegi per pantalla i l'executa en el background directament.

**jobs:** diu els processos que s'estan executant.

**fg %<numero>:** puja a primer pla el procés amb aquell número.

**kill %<numero>:** mata el procés amb aquell número.

**ps -ef:** treu una snapshot de tots els processos.

**pidof "procés":** treu el PID dels processos.

root	8602	2	0	12:42	?	00:00:00	[kworker/u6:2-events_power_efficient]
root	8607	2	0	12:47	?	00:00:00	[kworker/u5:1-events_unbound]
pau	8623	1098	6	12:49	pts/0	00:00:00	ps -ef

(1) qui ha creat el process

(2) identificador del proces (pid)

(3) lloc d'on s'ha creat el proces (ppid)

Com treure els ppid dels processos.

**ps -ef h | awk '{print \$3}' | sort -n | uniq**

**date +%F: YYYY-MM-DD**

**date +%R: hh:mm**