5.1. Automatització

Yolanda Alemany Ruiz 2020-2021

Introducció

Tots els intèrprets de comandes incorporen un llenguatge de programació que inclou:

- Sentències de control de flux d'execució
- Creació i assignació de variables

Els usuaris poden escriure programes (ShellScripts) per tal d'automatitzar l'execució de seqüències de comandes.

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**

Exercici 1

Crea un shellscript que mostri la cadena de caràcters: “Hello, world!”

Per editar pots utilitzar: nano, vi o interfície gràfica Gedit.

Executa el shellscript des de la línia de comandes.

Passos bàsics per a crear i executar un shellscript

- Per indicar comentaris a un shellscript ho fem amb **#**.
- Per a buscar informació sobre com crear shellscripts, consultem el manual de l'intèrpret: **man bash**.

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)
```

Important: L'assignació de valors a variables no porta cap espai (**=**).

Definició de variables

- 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
```

Les variables als shellscripts **NO s'han de declarar** NI cal assignar un tipus prèviament al seu ús.

Cometes

- 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
```

Comandes útils (2)

- **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
```

Comandes útils (3)

- **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 lim; 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
```

Exercici 2

Crea les variables a=5, b=65, c=75.

Fes les següents operacions:

- d = a + b + c
- e = a ^ 3 + c
- f = e mod a
- g = 7 + 30
- h = g + 1
- i = valor introduït per teclat + 3

Mostra el resultat de **d**, **e**, **f**, **g** i **h**.

Comandes útils (3)

- **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)
```

Comandes útils (3)

- **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
```

Exercici 3

Crea un shellscript que, a partir d'un nombre decimal introduït per teclat, et mostri el missatge "El nombre NUM és parell" o "El nombre NUM és senar", segons si és senar o parell.

Exercici 4

Modifica el shellscript anterior però fes que l'usuari pugui anar introduint nombres i obtenir resultat.

> Introdueix un nombre enter:

> El nombre NUM és senar.

> Introduexi un nombre enter:

....

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: **for**

NOTA: molta cura amb l'ús de les cometes dobles ("").

```
1  List="one two three"
2
3  for a in $List # splits the variable in parts at whitespace
4  do
5       echo "$a"
6  done
7  # one
8  # two
9  # three
10
11  echo "---"
12
13  for a in "$List" # preserves whitespace in a single variable
14  do #
15       echo "$a"
16  done
17  # one two three
```

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
```

Control del flux d'execució: consideracions finals

Dins els bucles **for, while** i **until** es poden utilitzar les sentències.

- **break**: fa que se surti del bucle.
- **continue**: fa que es passi a la següent iteració del bucle.

Oblidar alguna de les paraules clau d'una sentència de control del flux d'execució (per exemple, **done**) o no tancar una cometa, pot provocar l'error **Unexpected EOF**.

Pas de paràmetres als shellscripts

Quan invoquem un shellscript, li podem passar una sèrie de paràmetres.

Els paràmetres són accessibles des de l'interior del shellscript:

- **$#**: conté el nombre de paràmetres d'aquell shellscript.
- **$***: és la llista de tots els paràmetres del shellscript (útil per a iterar).
- El primer paràmetres és accessible amb **$1**, el segon amb **$2**,... el novè amb **$9** i a partir del desè cal indicat **${10}**, **${11}**...
- La variable **$0** conté sempre el nom del shellscript.
- **$?**: codi d'acabament de l'última comanda executada.
- Amb la variable **$$** podem consultar el PID del procés del shellscript.

Quan un shellscript espera paràmetres, és aconsellable comprovar que el nombre de paràmetres (**$#**) sigui l'esperat.

Si volem que el shellscript retorni un codi d'acabament utilitzem la comanda **exit**, seguida d'un codi numèric d'acabament (**0 cert/OK**).

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.

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.

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 7

Crea un shellscript que demana la ruta absoluta d'un fitxer i indiqui si és un directori, un fitxer regular o un enllaç.

Afegeix control d'errors!

Exercici 8

Crea un shellscript que faci la funció de multiplicació però sense utilitzar l'operador corresponent (·). L'usuari ha de passar com a paràmetres dos nombres.

Recorda fer el control d'errors.

Exercici 9

Crea un shellscript anomenat “op.sh”. L’usuari ha d’introduir dos paràmetres numèrics i un tercer paràmetre que pot ser “*”, “+”, “-” o “mod”. Segons aquest tercer paràmetre, es faran les operacions corresponents a cada símbol i es retornarà el resultat.