Guia bàsica de Bash

Introducció

Aquesta guia descriu algunes eines bàsiques per a la programació d'scripts amb Bash. Per a una informació més detallada podeu anar a les guies descrites a la bibliografia o a la pàgina de manual.

1. Execució d'scripts

Script d'exemple per començar [1]

 # cat sistema.sh

```
#!/bin/bash

clear

echo “informacio basica del sistema”

echo “Hola, $USER”

echo “La data de avui es `date`”

echo “Aquests usuaris estan connectats ara mateix:”

w | cut -d " " -f 1 | grep -v USER | sort -u

echo

echo “Aquest es un sistema `uname -s` a un processador `uname -m`”

echo

echo “informació del estat actual del sistema”

uptime

echo

echo “informació de la memora lliure:”

free
```

- Com executar l'script

  # chmod u+x sistema.sh

  # ./sistema.sh

- Debug de l'script

  # bash -x sistema.sh

2. L'entorn de Bash

2.1 Fitxers de Configuració

- Fitxer de configuració global de variables com: PATH, USER, MAIL. etc

  /etc/profile

- Fitxer de configuració individual: pels shells amb *login*

  ~/.bash_profile

- Fitxer de configuració individual pels shells amb accés sense *login*

  ~/.bashrc

- Fitxer que se executa al sortir de Bash (*logout*)

  ~/.bash_logout

2.2 Prompt

- A l'execució interactiva Bash mostra la cadena “prompt” que està definida a la variable PS1

  # echo $PS1

- (Algunes) Seqüències d'escapament:

  \d: data

  \h: hostname

  \u: nom d'usuari

  \w: directori de treball actual

  \$: mostra # si el UID de l'usuari és zero, en altres casos representa $

3. Variables

3.1 Crear i exportar variables

- El nom d'una variable pot tenir lletres i números però no començar amb números.
- Mostra totes les variables globals:

  # printenv

- Creació d'una variable:

  # VARNAME=”value”

- **ATENCIÓ:** Posar espais abans o després del signe “=” dona errors.
- Exportar variables: necessari quan volem que el valor d'una variable sigui visible per un subshell o programa executat pel shell.

  # export VARNAME

- Es pot definir i exportar una variable en una única comanda:

  # export VARNAME=”vaule”

3.2 Variables predefinides (una llista reduïda de les més usades)

- HOME: directori home de l'usuari actual.
- HOSTNAME: el nom de la maquina local.
- OLDPWD: directori de treball previ.
- PATH: llista de directoris on el shell cerca les comandes.
- PS1: cadena principal de *prompt.*
- PWD: directori de treball actual.
- SHELL: nom del shell actual.

3.3 Variables especials

Són valors especials que només poden ser llegits, mai sobreescrits:

- $#: nombre de paràmetres passats a l'script
- $0: nom del shell script
- $n: (n>0) valor del paràmetre n passat a l'script
- $*: fa una expansió de la llista de paràmetres passats a l'script
- $?: resultat de l'execució de la comanda anterior

4. Les cometes

Com que alguns caràcters tenen significats especials és necessari utilitzar cometes o contrabarres quan vulguem eliminar aquest significats.

4.1 Caràcter d'escapament

La contrabarra \ permet preservar el valor literal del següent caràcter.

```
$ VAR=hola,\ $USER

$ echo $VAR

# hola alvarez
```

4.2 Cometes senzilles

Preserven el valor literal dels caràcters entre les cometes

```
$ VAR='hola,\ $USER'

$ echo $VAR

hola,\ $USER
```

4.3 Cometes dobles

Preserven el valor literal dels caràcters entre cometes, excepte el signe de dolar ($), les cometes senzilles invertides (` `) i la contrabarra (\)

```
$ VAR=”hola, $USER. Avui es `date +%F`”
```

```
$ echo $VAR

hola, alvarez. Avui es 2008-01-10
```

5. Expansió del Shell

Després de fer el parsing de les paraules d'un script el Shell fa diferents expansions i transforma els *tokens* en una llista de noms de fitxer, comandes i arguments. Les següents són les expansions més comuns:

5.1 Expansió de {}

Permet generar múltiples cadenes de text

```
# mkdir -p {home1,home2}/{src,bin,lib}
```

5.2 Substitució de comandes

La substitució de comandes permet a la sortida d'una comanda substituir a la comanda mateixa. Hi ha dues formes de substituir comandes:

- Utilitzant les cometes simples invertides:

  ```
  # touch `date`.log
  ```

- Utilitzant el signe dolar i els parèntesis :

  ```
  # touch $(date).log
  ```

5.3 Avaluació Aritmètica

Permet fer l'avaluació d'una expressió aritmètica i substituir-la pel resultat. L'avaluació aritmètica es pot fer amb la funció **let** o també si fem sevir la comanda **$((EXPRESSIO))**. Els operadors disponibles són:

- + - : suma i resta
- VAR++ VAR-- : postincrement i postdecrement
- ++VAR --VAR : preincrement i predecrement
- ! i ~ : negació lògica i de bits
- ** : potenciació
- *, /, % : multiplicació, divisió, mòdul
- << >> : desplaçament a la esquerra i a la dreta
- < <= > >= : comparacions
- == != : mateix , diferent
- & : operació AND de bits
- && : operació AND lògica
- | : operació OR de bits
- || : operació OR lògica
- ^ : operació XOR de bits
- expr?expr:expr : avaluació condicional

- =, +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>= : operadors de assignació

5.4 Expansió de titlla

- ~ : substitueix $HOME
- ~+ : substitueix $PWD
- ~- : substitueix $OLDPWD

 # export PATH="$PATH:~/bin"

5.5 Substitució de variables

- $VAR : referència a la variable VAR
- ${VAR} : referència a la variable VAR amb delimitació del nom
- ${VAR[n]} : element n del array VAR
- ${#VAR} : grandària de la variable VAR (en bytes)
- ${#VAR[*]} : nombre de elements dins l'array VAR
- ${#VAR[@]} : nombre de elements dins l'array VAR
- ${VAR#pat} : talla la subcadena més curta des del principi de VAR que coincideix amb pat
- ${VAR##pat} : talla la subcadena més llarga des del principi de VAR que coincideix amb *pat*
- ${VAR%pat} : talla la subcadena més curta des del final de VAR que coincideix amb *pat*
- ${VAR%%pat} : talla la subcadena més llarga des del final de VAR que coincideix amb *pat*
- ${VAR/pat/} : el primer valor de VAR que coincideix amb *pat* es esborrat
- ${VAR//pat/cad} : tots els valors de VAR que coincideix amb *pat* són reemplaçats per *cad*
- ${VAR/#pat/cad} : el valor al principi de VAR que coincideix amb pat es reemplaçat per cad
- ${VAR/%pat/cad} : el valor al final de VAR que coincideix amb pat es reemplaçat per cad

Exemple

```
$ VAR=cadenadetext.tar.gz

$ echo ${VAR##*de}

text.tar.gz

$ echo ${VAR#*de}

nadetext.tar.gz

$ echo ${VAR%%.*}

cadenadetext

$ echo ${VAR%.*}

cadenadetext.tar
```

```
$ echo ${VAR/text*}

cadenade

$ echo ${VAR/#ca/ra}

radenadetext.tar.gz
```

6. Expressions regulars

6.1 Introducció a les expressions regulars

- Les expressions regulars permeten descriure de manera compacta conjunts de cadenes de caràcters.
- Els elements bàsics de les expressions regulars són aquells que permeten descriure un sol caràcter.
- Les lletres i els dígits són expressions regulars que els descriuen a ells mateixos.

Exemple

```
$ date

sáb ene 1 18:59:46 CET 2008

$ date | grep -q 2008

$ echo $?

0

$ date | grep -q 2007

$ echo $?

1
```

6.2 Operadors de expressions regulars (Metacaràcters)

Una expressió regular es pot construir amb operacions sobre caràcters regulars utilitzant els operadors definits:

- . : qualsevol caràcter (però només per a un)
- ? : el ítem precedent al menys una vegada
- * : el ítem precedent zero o més vegades
- + : el ítem precedent una o més vegades
- {N} : el ítem precedent N vegades
- {N,} : el ítem precedent N o més vegades
- {N,M} : el ítem precedent al menys N vegades i com màxim M
- - : defineix un rang
- ^ : principi de la línia

- $: final de la línia
- \< : principi de paraula
- \> : final de paraula
- \b : límit de paraula (principi o final)
- \B : no límit de paraula

Exemple

```
$ grep ^bin /etc/passwd

bin:x:2:2:bin:/bin:/bin/sh

bind:x:114:123::/var/cache/bind:/bin/false

grep '^bin\>' /etc/passwd

bin:x:2:2:bin:/bin:/bin/sh
```

Exemple

```
$ word1=1980c

$ word2=1980

$ echo $word1 | grep -q '^[0-9]\+[a-d]$'

$ echo $?

0

$ echo $word2 | grep -q '^[0-9]\+[a-d]$'

$ echo $?

1
```

6.3 Classes de caràcters

- Una expressió entre claudàtors ([]) representa una llista de caràcters i es fa concordança amb qualsevol caràcter de la llista.
- Hi ha una llista de classes de caràcters especials definits per el estàndard POSIX:
  - [:alnum:] : caràcters alfanumerics
  - [:alpha:] : caràcters alfabètics
  - [:blank:] : caràcters d'espai: espai en blanc i tabulador
  - [:digit:] : equivalent a [0123456789]
  - [:word:] : caràcters de paraula

Exemple

```
$ df | tr -s [:blank:] | cut -d " " -f4
```

```
Available

231520

249320
```

7. Redirecció de entrada i sortida

- comanda < fitxer : utilitza *fitxer* com entrada
- comanda > fitxer : utilitza *fitxer* com sortida sobreescrivint
- comanda >> fitxer : utilitza *fitxer* com sortida afegint
- comanda &> fitxer : envia *stdout* i *stderr* a fitxer
- comanda >& fitxer : envia *stdout* i *stderr* a fitxer
- comanda 2>&1 > fitxer : envia *stdout* i *stderr* a fitxer

8. condicionals

8.1 if

if LLISTA-1; then LLISTA-2; [elif LLISTA-3; then LLISTA-4]; [else LLISTA-5] fi

Exemple

```
!/bin/bash

word=${1}

pattern='^[0-9]\+[a-d]$'

if echo $word | grep -q $pattern

then

  number=`echo $word | grep -o '^[0-9]\+'`

  letter=`echo $word | grep -o '[dma]$'`

  echo "number=$number, letter=$letter"

else

  echo "$pattern not found in $word"

fi
```

8.2 Expressions condicionals: [expressió]

- cadena : verdader si la cadena no és NULL
- -a fitxer : verdader si el fitxer existeix
- -d fitxer : verdader si fitxer es un un directori
- -e fitxer : verdader si el fitxer existeix

- -f fitxer : verdader si és un fitxer regular
- -h fitxer : verdader si fitxer es un enllaç simbòlic
- -n cadena : verdader si cadena té una longitud diferent de zero
- -w fitxer : verdader si fitxer es pot escriure en fitxer
- -x fitxer : verdader si fitxer es pot executar el fitxer
- -z cadena : verdader si cadena té longitud zero
- fitxer1 -nt fitxer2 : verdader si fitxer1 és més nou que fitxer2
- fitxer1 -ot fitxer2 : verdader si fitxer1 es mes antic que fitxer2
- cadena1 == cadena2 : verdader si cadena1 i cadena2 són iguals
- cadena1 != cadena2 : verdader si cadena1 i cadena2 no són iguals
- cadena1 < cadena2 : verdader si cadena1 és abans de cadena2
- cadena1 > cadena2 : verdader si cadena1 és després de cadena2

8.3 Expressions condicionals amb operadors aritmètics.

- arg1 -eq arg2 : verdader si arg1 és igual a arg2
- arg1 -ne arg2 : verdader si arg1 és diferent de arg2
- arg1 -lt arg2 : verdader si arg1 és menor que arg2
- arg1 -le arg2 : verdader si arg1 és menor o igual que arg2
- arg1 -gt arg2 : verdader si arg1 és major que arg2
- arg1 -ge arg2 : verdader si arg1 és major o igual que arg2

8.4 Altres expressions

- (expressió) : verdader si la expressió és verdadera
- !expressió : verdader si la expressió és falsa
- exp1 || exp2 : verdader si una de les expressions és verdadera
- exp1 && exp2 : verdader si les dos expressions són verdaderes

Exemple

```
#!/bin/bash

$ filename=$1

if [ -f $filename ]; then

    echo “propietats del fitxer:”

    echo “grandària: `ls -lh $filename | awk '{print $5 }'`“

    echo “tipus: `file $filename | cut -d”:” -f2`

else
```

```
    echo “el fitxer $filename no existeix”
fi
```

ATENCIÓ: s'ha de posar un espai abans i després del primer claudàtor i un altre abans del segon.

8.5 sentencia CASE

- Sintaxis:

  case EXP-1 in case1) COMANDES-1;; case2) COMANDES-2;; ... caseN) COMANDES-N;; esac

Exemple

```
case $1 in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        ;;
esac
```

9. Tasques repetitives

9.1 bucle FOR

- **for VAR [in LLISTA]; do COMMANDES; done**

Exemple

```
for user in `cat /etc/passwd | cut -d: -f1` do
    echo $user
done
```

9.2 Bucle WHILE

- **while COMMANDES-1; do COMMANDES-2; done**

Exemple

```
i=”0”

while [ $i -lt 5 ]

    echo $i

    $i=[$i+1]

done
```

9.3 bucle UNTIL

- Sintaxis: **until COMMANDES-1; do COMMANDES-2; done**

10. Built-ins:

Els built-ins són comandes internes de Bash que es poden fer servir dintre dels scripts. A continuació presentem una llista dels més utilitzats:

- **source fitxer** : llegeix i executa comandes d'un fitxer
- **break** : surt d'un bucle for, while o until
- **cd** : canvia el directori actual
- **continue** : fa la següent iteració d'un buble for, while o until
- **echo** : mostra un missatge per la sortida estàndard
- **exit [n]** : surt de l'script amb el codi n
- **export nom=val** : exporta la variable *nom* amb el valor *val*
- **kill jobid** : envia la senyal SIGTERM a un procés
- **let expr** : avalua l'expressió aritmètica
- **pwd** : imprimeix el directori actual
- **read** : llegeix de l'entrada estàndard
- **return [n]** : surt de l'script amb el codi n
- **shift [n]** : renombra els paràmetres d'entrada.
- **test** : avalua una expressió condicional
- **wait [jobid]** : espera que el procés termini

11. Altres utilitats

11.1 Noms de fitxer i de directori

- Extreure el nom de fitxer o de directoris
  # basename /etc/apt/sources.list
  # sources.list

- Extreure el nom del directori
  # dirname /etc/apt/sources.list
  # /etc/apt

11.2 Gestió de la data

- Imprimir la data en format: “dia mes hora:minut:segon”
  # date
- Imprimir el nombre de segons des de la “UNIX epoch” (01/01/1970). Molt útil per fer comparacions de dates.
  # date +%s
- Fer conversions de dates
  # date -d

Exemple

```
$ date

Fri Oct 26 19:41:29 CEST 2007

$ date  +%s

1193420527

$ date -d “Dic 31 2007 23:59:59” +%s

1199141999

if [ `date +%s` -gt `date -d "Dec 31 2007 23:59:59" +%s` ] ; then

    echo "Ara som al 2008";

fi
```

11. Referències Bibliogràfiques

[1] M. Garrels. **Bash Guide for Beginners**. Online: The Linux Documentation Project.

http://tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf

[2] D. Robbins, **Bash by example**. Online: IBM Developer Works

http://www-128.ibm.com/developerworks/linux/library/l-bash.html?ca=drs-

[3] Arnold Robbins. **BASH Reference Card**. Online: http://www.scc.com