

Introducció



es comencen amb: `#!/bin/bash`

/alt
interpret

`for i in "1,2,3,4": do`

`echo $i`

`- ls -l & .jpg;`
`$ (ls -l & jpg)`

done

`JPG = 'ls -l & -jpg'`

`function xxx() {`

`AAA = 3 → quasi equivalent a for la global`

Introducció a Perl

```
#!/usr/bin/perl  
use strict → ens ajuda a fer les coses ben fetes
```

```
my $a; → declarar  
$a = 3;  
$a = "holà";  
$a = $a + 1; → error! No intentarà
```

} Escalars

```
my @b = [1, "holà", 3]  
$b[0]; → ningú treballa amb l'índex  
push @b, 3; → Afegix al final  
$a = pop @b; → Esborra la del final  
$a = shift @b → Esborra la del principi
```

ITERAR

```
for each $a (@b  
    print $a;  
{  
    $a eq "-P") {  
    $a == 3  
    !=  
    >=
```

} Array

```
my %h  
$h{"Bob"} = 1.73;  
$h{"Alice"} = 1.84;  
  
if !$h{"Alice"} {  
}  
for each $c (keys %h)  
{  
    $h{$key}
```

} Hash
(diccionaris)

Pràctica 3: Scripts

Introducció

Normalment les tasques d'administració han de repetir-se una vegada i una altra, motiu pel qual l'administrador ha d'escriure de nou les comandes i en ocasions canviant només algun paràmetre d'entrada. Fer aquestes tasques manualment no només implica una inversió considerable de temps sinó que exposa el sistema a errors quan es repeteix una comanda de forma equivocada. L'automatització d'aquestes tasques mitjançant llenguatges d'script millora l'eficiència del sistema ja que aquestes es realitzen sense la intervenció de l'administrador. També augmenta la fiabilitat perquè els comandes es repeteixen de la mateixa forma cada vegada i a més a més permet garantir la regularitat en la seva execució perquè aquestes tasques es poden programar fàcilment perquè s'executin periòdicament.

Encara que l'automatització es podria fer en qualsevol llenguatge de programació existeixen llenguatges, coneguts com llenguatges d'scripting. Aquest llenguatges permeten combinar fàcilment expressions del propi llenguatge d'script amb comandes del sistema, i també faciliten la manipulació de fitxers de text, llistes, i el recorregut i tractament de directoris, i altres tasques útils per a l'administració. Existeixen múltiples llenguatges d'scripting: els associats al shell (com Bash o C shell) i altres amb funcionalitats més esteses com el Perl o el Python.

Objectius

Aprendre a automatitzar algunes tasques comunes d'administració de sistemes fent ús de llenguatges d'script com el Bash i el Perl

Abans de començar

- Repassar la programació de shell-scripts amb Bash
- Analitzar les construccions bàsics del llenguatge Perl.

La pràctica consisteix en dues parts. En la primera s'analitzarà un script d'exemple per a la detecció d'usuaris “innecessaris” en el sistema. En la segona part es proposa fer un script per a la gestió l'espai en disc. Els scripts es poden fer en Bash o en Perl. A més al final es proposa una llista de problemes per practicar fora del laboratori.

Tingueu en compte en tot moment les propietats d'un bon script [1]:

- 1) Un script ha d'executar-se sense errors.
- 2) Ha de realitzar la tasca per a la qual està pensat.
- 3) La lògica del programa ha d'estar clarament definida.

- 4) Un script no ha de fer treball innecessari.
- 5) Els scripts han de ser reutilitzables.

Script d'exemple: Usuaris innecessaris

Es demana fer un script que determini quins usuaris del fitxer /etc/password són invàlids. Un usuari invàlid és aquell que existeix en el fitxer de passwd però que en canvi no té cap presència en el sistema (és a dir, que no té cap fitxer). També, hi ha usuaris que no tenen cap fitxer, però que serveixen per executar daemons del sistema. Afegiu una opció per declarar vàlids als usuaris que tenen algun procés en execució (flag -p).

Exemple de la sortida:

```
$./BadUsers.pl
```

```
daemon
```

```
bin
```

```
sys
```

```
sync
```

```
games
```

```
lp
```

```
mail
```

```
news
```

```
aduran
```

```
alvarez
```

```
proxy
```

```
backup
```

```
$./BadUsers.pl -p
```

```
bin
```

```
sync
```

```
games
```

```
lp  
news  
aduran  
alvarez  
proxy  
backup
```

El script fet en Perl

Completeu els espais buits amb les sentencies necessàries en cada cas.

Paràmetres d'entrada

En primer lloc detectarem l'opció d'entrada i farem un control d'errors. Només acceptarem un paràmetre d'entrada que sigui “-p”

```
#!/usr/bin/perl  
  
$numArgs = @ARGV;  
  
$p=0;  
  
$usage="Usage: BadUsers.pl [-p] \n";  
  
if ( $numArgs != 0){  
    if ( $numArgs == 1 ){  
        if ( ARGV[0] eq "-p" ) { # si el primer argument es -p  
            $p=1; # activem el flag p  
        } else { print $usage; exit (1); }  
    } else { print $usage; exit (1); }  
}
```

Llegir el fitxer de passwd

Ara farem que l'script obri la base de dades d'usuaris i que guardi totes les dades dels usuaris en una llista.

```
$pass_db_file="/etc/passwd";  
  
open (FILE,$pass_db_file) or die "no es pot obrir el fitxer $pass_db_file: $!";  
@password_db= <FILE>; # llegir tot el fitxer de password  
close FILE;
```

Què vol dir exactament aquesta sentència?

`open(FILE,$pass_db_file) or die "no es pot obrir el fitxer $pass_db_file: $!";`

obrir el fitxer o posar la exec del programa.

Recorregut de la llista d'usuaris

Ara recorrerem la llista d'usuaris i buscarem per a cadascun si tenen fitxers al sistema tot usant la comanda `find` (veure `man find`). El usuaris sense fitxers els guardarem a un `hash` com invàlids per després utilitzar-los per processar la opció `-p`.

```
foreach $user_line (@password_db) {  
    chomp($user_line); # eliminar el salt de línia  
  
    @fields = split(':', $user_line);  
  
    $user_id = $fields[0];  
  
    $user_home = $fields[5];  
  
    if ( -d $user_home ) {  
  
        $comand=sprintf("find %s -type f -user %s | wc -l",  
                        $user_home, $user_id);  
  
        $find_out=`$comand`;  
        chomp($find_out); # eliminar el salt de línia  
  
    } else {  
  
        $find_out = 0;  
  
    }  
  
    if ($find_out == 0){  
  
        $invalid_users{$user_id} = "invalid";  
  
    }  
}
```

Quin és el significat dels paràmetres de la comanda `find`?

`$comand=sprintf("find %s -type f -user %s | wc -l", $user_home, $user_id);`

printpe els usuaris

Quina és la diferència entre les funcions `chomp` y `chop`?

`chomp` elimina els caràcters del final d'un string corresponents a un separador de línia i `chop` elimina directament l'últim caràcter

Quin és el significat de la sentència?

`$invalid_users{$user_id} = "invalid";`

Assignarà invàlid al element de la llista
\$user_id

Usuaris que tenen processos en execució

Ara afegirem l'opció per no incloure com invàlids al usuaris que tenen processos en execució. Per trobar el usuaris que tenen processos en execució usarem la comanda `ps aux --no-headers` (veure `man ps`) i els eliminarem del *hash* de usuaris invàlids. Els usuaris que queden a la llista com invàlids seran aquells que no tenen cap fitxer ni procés en execució.

Utilitzeu una expressió regular com delimitador dels camps de la sortida de la comanda `ps`. Tingueu en compte que us podeu trobar amb un o més espais buits i amb tabuladors.

```
if ( $p == 1 ){

    @process_list=`ps aux --no-headers`;

    foreach $process_list_line (@process_list) {

        chomp($process_list_line);

        @fields_proc = split(":", "$process_list_line");

        $user_proc = $fields_proc[0];

        delete($invalid_users{$user_proc});

    }

}

foreach $user_inv_id (sort((keys%invalid_users))){  
    print "$user_inv_id\n";
}
```

Què fa exactament la comanda `delete($invalid_users{$user_proc});`

elimina del hash el usuari

i la comanda: `sort(keys%invalid_users)`

ordene alphabeticalment els noms

El script en Bash

Ara teniu el mateix script fet en Bash. Completeu els espais buits amb les comandes apropriades.

```
#!/bin/bash

p=0

usage="Usage: BadUser.sh [-p]"

# detecció de opcions d'entrada: només son vàlids: sense paràmetres i -p

if [ $# -ne 0 ]; then
    if [ $# -eq 1 ]; then
        if [ $1 == "-p" ]; then
            p=1
        else
            echo $usage; exit 1
        fi
    else
        echo $usage; exit 1
    fi
fi

# afegiu una comanda per llegir el fitxer de password i només agafar el camp de # nom de l'usuari
for user in `cat -d : -f /etc/passwd`; do
    home=`cat /etc/passwd | grep "^$user>" | cut -d: -f6`
```

```

if [ -d $home ]; then
    num_fich=`find $home -type f -user $user | wc -l`
else
    num_fich=0
fi

if [ $num_fich -eq 0 ] ; then
    if [ $p -eq 1 ]; then

# afegiu una comanda per detectar si l'usuari té processos en execució,
# si no té ningú la variable $user_proc ha de ser 0
    user_proc=`ps -u "$user" | wc -l` . . .

if [ $user_proc -eq 0 ]; then
    echo "$user"
    fi
else
    echo "$user"
    fi
fi
done

```

Què vol dir exactament aquesta comanda:
`cat /etc/passwd | grep "^\\$user>" | cut -d: -f6`?

Quina diferència hi ha amb la comanda:
`cat /etc/passwd | grep "\$user" | cut -d: -f6`?
^{^ fa que es coloqui el principi de la línia.}

Detecció de usuaris inactius

Ara estereu aquest script per detectar usuaris *inactius*. Es defineix com

Accessos a la varxa i al shell

Octubre per defecte

inactius als que no executen cap procés (veure comanda `ps` al punt anterior), que fa molt de temps que no han fet login (ver comandes **finger**, **last** i **lastlog**), i que fa molt de temps que no han modificat cap dels seus fitxers (veure opcions `time` a la comanda `find`). El període d'inactivitat s'indicarà a través d'un paràmetre:

```
$./BadUsers.pl -t 2d (indica 2 dies)
```

```
alvarez
```

```
aduran
```

```
xavim
```

```
marcg
```

```
$./BadUsers.pl -t 4m (indica 4 mesos)
```

```
xavim
```

```
marcg
```

Indiqueu les modificacions necessàries per donar suport a la nova opció d'usuaris inactius

```
t1=$1
t2=$2
if [ "$t2" == "m" ]; then
    t3=${t1}month
    t4=$(( ${t1} * 30 ))
fi

if [ "$t2" == "y" ]; then
    t3=${t1}year
    t4=$(( ${t1} * 365 ))
fi

echo "t = ${t}"
echo "p = ${p}"
echo "t1 = ${t1}"
echo "t3 = ${t3}"

# afegiu una comanda per llegir el fitxer de password i només agafar el camp de # nom de l'usuari
for user in `cut -d : -f 1 /etc/passwd`; do
    home=`cat /etc/passwd | grep "^\$user\>" | cut -d: -f6`
    if [ -d $home ]; then
        num_fich=`find $home -type f -user $user | wc -l`
    else
        num_fich=0
    fi
    if [ $num_fich -eq 0 ] ; then
        if [ $p -eq 1 ]; then
            echo " "
        fi
    # afegiu una comanda per detectar si l'usuari té processos en execució,
    # si no té ningú la variable $user_proc ha de ser 0
        user_proc=`ps -u $user -o comm= | wc -l`
        if [ $user_proc -eq 0 ]; then
            echo "$user"
        fi
    else
        echo "$user"
    fi
    fi
    if [ $ti -eq 1 ]; then
        user_proc=`ps -u $user -o comm= | wc -l`
        if [ $user_proc -eq 0 ]; then
            ultimlog=`last -f /var/log/wtmp.1 -s $t3 | grep $user`
            if [[ $ultimlog -eq 0 ]]; then
                ultimmodif=`find $home -type f -user $user -mtime $t4`
                if [[ $ultimmodif -eq 0 ]]; then
                    echo "$user"
                fi
            fi
        fi
    fi
done
```

Script per a la gestió de l'espai en disc

S'ha de fer un script que calculi l'espai en disc utilitzat per cada usuari del sistema. Si sobrepassa un espai determinat que es passa com paràmetre, s'haurà d'escriure un missatge al *.profile* del usuari en qüestió per informar-li que ha d'esborrar/comprimir alguns dels seus fitxers.

Concretament , la sintaxi del programa ha de ser la següent:

```
$ ocupacio.sh max_permès
```

Per exemple:

```
$ ./ocupacio.sh 600M
root      567 MB
alvarez   128 KB
aduran    120 MB
xavim     23 MB
( ... )
```

Després esteneu el script per afegir una opció per grups **-g**: Amb aquesta opció l'script ha de retornar l'ocupació total per als usuaris del grup <grup>, el total d'ocupació del grup sencer, i posar el missatge als usuaris que sobrepassen el max_permès.

Per tant, la sintaxi final del programa haurà de ser:

```
$ ocupacio.sh [-g grup] max_permès
```

Per exemple:

```
$ ./ocupacio.sh -g users 500K
alvarez   128 KB
xavim     23 MB
( ... )
```

NOTA: El missatge que s'ha de posar en el *.profile*, ha de poder ser localitzat i esborrat per l'usuari sense cap tipus de problema. Això vol dir que al costat del missatge s'haurien de posar instruccions per poder-lo esborrar sense cap problema.

Scripts per practicar

A continuació teniu una sèrie d'exercicis sobre scripts que podeu fer per practicar. Haurieu de saber fer-los tant en shell script com en Perl.

Informació d'usuari

Volem fer un script que donat un nom d'usuari ens doni la següent informació relacionada amb ell:

- Home
- Tamany total del directori home (tot incloent subdirectoris)
- Directoris fora del directori home on l'usuari té fitxers propis
- Nombre de processos actius de l'usuari

Una possible sortida seria:

```
$./infouser.sh aduran  
  
Home: /home/aduran  
  
Home size: 2.5G  
  
Other dirs: /tmp /home/common  
  
Active processes: 5
```

Estadístiques de login y processos

Volem fer un script (**user-stats**) que ens doni un resum de tots els accessos de tots els usuaris a la màquina. El resum ha d'incloure per a cada usuari del sistema el temps total de login y el nombre total de logins que cada usuari ha fet (veure comanda **last**). A més a més, per als usuaris que tinguin connexions actives es demana reportar el nombre de processos que tenen en execució i el percentatge de CPU que estan utilitzant (veure comanda **ps**).

La sortida de l'script ha de ser similar a :

```
./user-stats.pl  
  
Resum de logins:  
  
Usuari aduran: temps total de login 115 min, nombre total de logins: 10  
Usuari marcg: temps total de login 153 min, nombre total de logins: 35  
  
  
Resum d'usuaris connectats  
  
Usuari alvarez: 3 processos -> 30 % CPU  
Usuari root: 10 processos -> 15% CPU
```

Estadístiques de comunicació

Volem fer un script que ens tregui la següent informació per a cada interfície de xarxa activa:

- Nom de la interfície: total de paquets transmesos

- Total: suma de tots els paquets transmesos en totes les interfícies actives

Per exemple:

```
./net-out

lo:    1621

wlan0:   64634

Total: 66255
```

Si ara volem que l'script vagi donant la informació en un terminal cada N segons, com ho faríeu? Suposeu que passem el temps d'espera per paràmetre a l'script, per exemple:

```
$ net-out 2 # amb una espera de 2 segons
```

Activitat dels usuaris

Volem classificar els usuaris de la màquina que administrem en funció de l'activitat que mostren en el sistema de fitxers. Realitzeu un script **class_act** que donat un nombre enter **n** i el nom i primer cognom d'un usuari (atenció, no es tracta del uid), ens informi del nombre de fitxers en el home de l'usuari, amb data de modificació entre la data actual i els n o menys dies anteriors, i l'espai que ocupen a disc.

Exemple:

```
$ ./class_act.sh 3 "Alex Duran"

Alex Duran (aduran) 150 fitxers modificats que ocupen 1.2 MB
```

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] Doug Sheppard. **Beginner's Introduction to Perl**.

<http://www.perl.com/pub/a/2000/10/begperl1.html>

[4] **perlintro -- a brief introduction and overview of Perl**

<http://perldoc.perl.org/perlintro.html>

[5] Randal L. Schwartz, Tom Phoenix, brian d foy, **Learning Perl, Fourth Edition**, July 2005. O'Reilly.