Pràctica 2: Guions en Bash

Autors: Aleix Segura Paz i Aniol Serrano Ortega

Sistemes Operatius

Grau en Enginyeria informàtica

Universitat de Lleida

Aleix Segura
Aniol Serrano

3.1. Comanda history

history és una comanda útil en la interacció amb l'intèrpret de comandes.

1. Cerqueu informació sobre aquesta comanda, expliqueu què fa i doneu exemples d'execució.

La comanda *history* en Linux és una eina '*built-in shell*' que serveix per visualitzar la llista de comandes utilitzades en la sessió de terminal. Permet als usuaris reutilitzar qualsevol comanda llistada sense haver de tornar a escriure-la.

Alguns exemples d'execució:

- Utilitzant *history* a seques obtenim la visualització de la llista de comandes usades des de l'inici de la sessió de terminal.
- Utilitzant *history* <*valor enter*> obtenim la visualització de la llista de comandes utilitzades però només les <*valor enter*> últimes.
- Per repetir una comanda podem utilitzar !1 per exemple per a tornar a executar la primera comanda que visualitzaria *history*.
- Relacionant amb pipes podem per exemple utilitzar *history* | *grep* <*string a buscar*> per visualitzar aquelles comandes de la sessió de terminal que contenen <*string a buscar*>.

Associada a la comanda *history* hi ha la variable HISTTIMEFORMAT:

2. Cerqueu informació sobre aquesta variable. De quin tipus és? Per a què serveix?

La variable HISTTIMEFORMAT és de tipus string - date. És una variable que serveix per a que quan utilitzem la comanda *history* puguem saber en quina data i en quin moment exacte vam utilitzar una comanda en concret.

3. Modifiqueu un dels fitxers d'arrancada del shell per establir-la a tots els vostres intèrprets de comandes. Quin fitxer heu modificat? Amb quin format de data-hora heu establert que surti la data de l'històric? Mostreu la comanda usada.

```
aleixse@aleixse:~$ echo 'export HISTTIMEFORMAT="%d/%m/%y %T "' >> ~/.bash_profile
aleixse@aleixse:~$ source ~/.bash_profile
```

```
  103  26/11/22 13:23:31 clear
  104  26/11/22 13:23:40 echo 'export HISTTIMEFORMAT="%d/%m/%y %T "' >> ~/.bash_profile
  105  26/11/22 13:23:50 source ~/.bash_profile
  106  26/11/22 13:23:53 history
aleixse@aleixse:~$
```

Hem modificat el fitxer bash_profile. Alternativament també s'hauria pogut modificar bash.rc. El format escollit ha estat dia-mes-any + hora complerta. Així ho indiquem amb "%d/%m/%y %T".

4. Aprofiteu aquesta modificació per establir al vostre PATH i de forma permanent (sense necessitat de repetir cap acció/comanda) el directori especial que fa referència al directori actual (.). Com ho heu fet? Indiqueu la comanda i el fitxer modificat.

```
aleixse@aleixse:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/g
ames:/snap/bin:/snap/bin
aleixse@aleixse:~$ PATH=~/bin/:$PATH:.
aleixse@aleixse:~$ echo $PATH
/home/aleixse/bin/:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr
/games:/usr/local/games:/snap/bin:/snap/bin:.
```

Simplement afegim a la variable PATH el directori actual.

3.2. Un primer script

Del següent script:

```
#!/dev/bash

if ( $# -ne 3 )
then
        echo "$0 suma els dos nombres passats com a parametres"
        echo "Ús: $0 <nombre1> <nombre2>"
        exit 1
end fi

echo "$1 + $2 = `expr $1 - $2`"
```

1. Copieu-lo al vostre entorn de treball, anomeneu-lo prac2_2.sh. Aquest guió té error/s de sintaxi, error/s de funcionament i/o error/s de compliment de requeriments. Cerqueu-los, quins són? Esmeneu-los fins que pugueu executar correctament l'script. Incloeu el script corregit en la resposta.

Llistat d'errors:

Línia 0: no es #!/dev/bash sinó #!/bin/bash

Línia 2: -ne s'ha de substituir per -eq i 3 s'ha de substituir per 2, ja que són el número d'arguments que requereix el nostre script. A més a més, també hem de comprovar que els paràmetres siguin dos enters. Ja que sinó, podríem passar per exemple un directori i un enter

i no arribaríem a veure l'ús correcte i tampoc ens faria òbviament la suma. Fiquem claudàtors ja que és un condicional més llarg.

Línia 3: cal indentar el *then*

Línia 5: l'ús hauria d'anar en la condició que s'hagi executat amb paràmetres incorrectes per tal que l'usuari pugui veure com s'ha d'executar correctament. Per tant, el posem dins d'un else afegint-hi al final un *exit* -1.

Línia 6: eliminem el *exit* 1.

Línia 7: eliminem el end

Línia 9: substituïm `expr $1 - $2 ` per `expr $(($1+$2))`.

Solució:

```
#!/bin/bash

if [[ $# -eq 2 && $1 =~ ^[0-9]+$ && $2 =~ ^[0-9]+$ ]]
        then
                echo "$0 suma els dos nombres passats com a parametres"
        else
                echo "Paràmetres incorrectes."
                echo "Ús: $0 <nombre1> <nombre2>"
                exit -1
fi

echo "$1 + $2 = `expr $(($1+$2))` "
```

2. A part de la identificació i correcció dels errors, indiqueu els passos que heu hagut de fer per a poder executar-lo (els passos més importants i fonamentals per a poder executar-lo).

Primerament, òbviament crear el script amb la comanda touch <nom_script>.sh al directori desitjat, seguidament hem copiat el contingut del script amb errors. L'hem corregit (important sobretot la primera línia del script que ha de ser #!/bin/bash) i després hagut d'otorgar permís d'execució al script amb chmod 0755 <nom_script>. Finalment l'execució del script es realitza de la mateixa forma que qualsevol altre executable: ./<nom_script>.sh <paràmetre1> <paràmetre2> o alternativament bash <nom_script>.sh <paràmetre1><paràmetre2> si no hem atorgat permís d'execució.

Aleix Segura
Aniol Serrano

3. Doneu un llistat de les variables que usa, indicant-ne el tipus al qual pertanyen i quina és la seva funcionalitat.

$# fa referencia al numero d'arguments, per tant de tipus enter.

$0 fa referencia al nom del script, per tant de tipus string.

$1 i $2 fan referencia al argument 1 i al argument 2, en aquest cas de tipus enter.

4. Expliqueu raonadament què fa.

Primer indiquem que es tracta d'un script de bash. S'avalua si el número d'arguments és igual a 2 i que els dos paràmetres passats siguin enters. En cas afirmatiu s'imprimeix per pantalla la seva funcionalitat i es realitza la suma de 3 formes diferents(`expr`, let i bucle). En cas negatiu imprimim per pantalla que els paràmetres són incorrectes, indiquem l'ús correcte del script i sortim retornant -1.

5. Afegiu 2 solucions alternatives a la darrera comanda del script mantenint la mateixa funcionalitat. És a dir, emprant altres mètodes de fer operacions aritmètiques. Entregueu aquesta versió del script.

```
#!/bin/bash

if [[ $# -eq 2 && $1 =~ ^[0-9]+$ && $2 =~ ^[0-9]+$ ]]
        then
                echo "$0 suma els dos nombres passats com a parametres"
        else
                echo "Paràmetres incorrectes."
                echo "Ús: $0 <nombre1> <nombre2>"
                exit -1
fi

echo "$1 + $2 = `expr $(($1+$2))` "

#Altres mètodes per fer la suma."
echo -e "\n"

let "a=$1" "b=$2"
let c=a+b
echo $c

echo -e "\n"

let "suma=0"
for i in $1 $2
do
         let "suma+=$i"
done

echo $suma
```

Aleix Segura
Aniol Serrano

Memòria Exercici 3.3

Hem dividit l'script en dos blocs. El primer bloc avalua si els paràmetres passats a l'script són vàlids o no. Si són vàlids guardarem en un *array* cada un dels usuaris i podrem avançar al segon bloc, sinó, indicarem per pantalla l'ús correcte de l'script.

Com sabem si els paràmetres són vàlids? S'han de complir una sèrie de condicions:

1. El nombre d'arguments ha de ser igual a 2. Ho aconseguim fent $# == 2.
2. El segon paràmetre té que ser un enter. Ho aconseguim fent: $2 =~ ^[0-9]+$.
3. La variable $HOME ha de contenir el directori on guardem els usuaris (primer paràmetre, "/home" en la majoria de casos). Ho aconseguim fent $HOME == *"$1"*.
4. Assegurem que el primer paràmetre sigui un directori. Ho aconseguim fent $1 == *"/"*.

Un cop hem validat els arguments, si són correctes avançarem al segon bloc.

En aquest bloc tenim una petita funció que guarda a la variable *capacity* el retorn de du -ms -x <directori> per a un usuari concret. A més a més, manipula el retorn per tal de guardar-ne només el valor numèric.

Per últim, tenim el nucli de l'script, on per a cada usuari que tenim dins l'*array* l'analitzem. Per a imprimir només el nom de l'usuari fem un tractament del valor /home/nom_usuari/ que tenim guardat a l'*array*. Seguidament, avaluem si aquell directori d'usuari és executable o no, és a dir, si en tenim accés. Si no en tenim mostrarem per pantalla que no podem llegir-lo, en canvi, si tenim accés obtenim la capacitat que té ocupada aquell usuari i avaluem si excedeix la quota de memòria passada per paràmetre o no. Un cop hem analitzat tots els usuaris acaba l'script.

Memòria Exercici 3.4

Primerament, hem analitzat el problema i l'hem dividit en dos casos, en el cas en què s'introdueixen paràmetres i en el que no. Quan passem paràmetres únicament hem d'analitzar els fitxers i/o directoris que es passen per paràmetre, per tant, podem crear un bucle *for* sense especificar on iterem (ja que per defecte en bash si no s'especifica en el bucle *for* es sobreentén que és sobre els paràmetres passats i així ho interpreta el programa. Únicament ja només ens cal identificar si el fitxer és vàlid o no, i si és un directori, en cas contrari retornem un error.

Per a identificar el tipus de fitxer usem la comanda *tipFitxer* que en si ja imprimeix el resultat. Ens fem servir de la comanda *file* i li passem com a argument el fitxer. En cas que la comanda *file* retorni un enter inferior a 0, és a dir, un retorni error printem un error en la comanda, altrament printem el nom i el tipus de fitxer.

Finalment, en el cas en què no passem cap paràmetre hem d'imprimir la ruta del directori junt amb el nom i tipus dels fitxers en aquella carpeta. Per a obtenir la ruta ho fem mitjançant la comanda *pwd* i per a obtenir el nom i tipus de fitxer ho fem exactament igual que en el cas on li passem els fitxers i/o directoris per paràmetres, però amb la diferència en què el bucle *for* l'iterem amb tots els continguts del directori.