SO

Seminar 2

Ce este un SHELL?

un interpretor de linie de comandă sau un mediu care furnizează o interfață utilizator în liniei de comandă pentru sistemele de operare similare Unix.
- limbaj interactiv de comandă
- limbaj de script
este utilizat de sistemul de operare pentru a controla execuția sistemului folosind scripturi shell.
Utilizatorii interacționează de obicei cu un shell Unix folosind un emulator de terminal; cu toate acestea, operarea directă prin conexiuni hardware seriale sau Secure Shell (ssh, vezi Putty) sunt comune pentru sistemele server.

Ce este un SHELL?

Toate shell-urile Unix furnizează:
- filename wildcarding
- piping
- here documents
- command substitution
- variabile și structuri de control pentru testarea condițiilor și iterare

În cel mai generic sens al termenului shell înseamnă orice program pe care utilizatorii îl utilizează pentru a tasta comenzi. Un shell ascunde detaliile sistemului de operare de bază și gestionează detaliile tehnice ale interfeței de kernel a sistemului de operare, care este cel mai scăzut nivel sau cel mai „interior” component al majorității sistemelor de operare.

Fisiere de comenzi

Un script (fişier de comenzi) este un fişier text care conţine în el:

- comenzi Unix;
- directive (ale interpretorului de comenzi shell) de control al fluxului execuţiei acestor comenzi.

Un script se comporta, la randul lui, ca si o comanda shell. Numele unui fişier script nu trebuie să respecte nici o cerinţă sintactică!

Noi vom adăuga la numele de scripturi sufixul .sh ca o convenţie proprie de a ilustra conţinutul.

Fisiere de comenzi

Orice comanda shell se poate rula:

1. Direct la prompter în linia de comanda
2. Comanda se scrie într-un script urmând a fi rulată odată cu execuţia scriptului

Dacă script este numele unui fişier de comenzi din directorul curent, rularea acestuia se poate face:

1. ./script ... sau caleabsoluta/script ... dacă fişierul script are drepturi de execuţie.

Pentru fixarea drepturilor, în particular şi a celor de execuţie, se foloseşte comanda chmod.

2. script ... dacă script are drepturi de execuție şi directorul curent este în PATH
3. sh script ... sau sh caleabsoluta/script ... indiferent dacă are sau nu drepturi de execuţie.

Prin ... am notat: argumente, optiuni, fisiere, expresii, redirectari: < > >> <& >&

SHELL Programming - Introducere

- De ce scriem scripturi?

- Viteza relativ la limbajul C

SHELL Programming - Intro

```
$ echo '#!/bin/sh' > my-script.sh

$ echo 'echo Hello World' >> my-script.sh

$ chmod 755 my-script.sh

$ ./my-script.sh

Hello World

$
```

SHELL Programming - Intro

```
#!/bin/bash

# This is a comment

echo Hello World                    # This is also a comment

-----------------------------------------------------

#!/bin/sh

# This is a comment

echo "Hello      World       "          # This is also a comment
```

SHELL Programming - Variabile

Nume simbolic pentru o zonă de memorie

putem atribui valori, putem sa citim sau sa modificam conținutul

Declarație prin atribuire: `VAR=valoare` funcționează; `VAR = valoarea` nu funcționează

Valoarea variabilei: `echo $VAR`

Variabile readonly: `VAR=„MyVar”`; readonly VAR

„Ștergerea” unei variabile: unset VAR

Variabilele readonly nu pot fi “șterse” (până nu se încheie procesul shell)

SHELL Programming - Variabile

Tipuri de variabile:

- **Variabile locale** − prezente în instanța actuală a shell-ului. Nu sunt disponibile pentru programele care sunt pornite de shell. Acestea sunt setate la linia de comandă
- **Variabile de mediu** − O variabilă de mediu este disponibilă oricărui proces copil shell. Unele programe au nevoie de variabile de mediu pentru a funcționa corect. De obicei, un script shell definește doar acele variabile de mediu care sunt necesare programelor pe care le rulează.
- **Variabile Shell** − O variabilă shell este o variabilă specială care este setată de shell și este necesară shell-ului pentru a funcționa corect. Unele dintre aceste variabile sunt variabile de mediu, în timp ce altele sunt variabile locale.

SHELL Programming - Variabile

Variabile speciale

$0 – numele comenzii care se ruleaza

$1 - $9 – argumentele liniei de comanda (în unele medii funcționează și ${10})

$* or $@ - toate argumentele

parametrul special „$ *” ia întreaga listă ca un argument cu spații între și parametrul special „$ @” ia întreaga listă și o separă în argumente separate.

$# - Numărul de argumente din linia de comandă

$? – codul de retur al comenzii anterioare

SHELL Programming - Variabile

```
#!/bin/sh

echo Who are you?

read ME

echo "Hello $ME , nice to meet you"
```

SHELL Programming - Variables

- Scope of variables
- Export

  $ VAR=hi

  $ ./script.sh

  $ export VAR

  $ ./script.sh

SHELL Programming - Variable Substitution

- Variable substitution permite programatorului sa manipuleze valoarea unei variabile pe baza stării sale.
- Posibile inlocuiri:
  - **${var}** Înlocuiește valoarea lui *var*.
  - **${var:-word}** Daca *var* este null sau unset, *word* inlocuieste **var**. Valoarea *var* nu se schimba.
  - **${var:=word}** Daca *var* este null sau unset, *var* este setat la valoarea **word**.
  - **${var:?message}** Daca *var* este null sau unset, *message* este afisat la standard error. Este util pentru a verifica setarea corecta a variabilelor.
  - **${var:+word}** Daca *var* este setat, *word* inlocuieste var. Valoarea lui *var* nu se schimba.

SHELL Programming - Variabile Sir (array)

parametrul special „$ *” ia întreaga listă ca un argument cu spații între și
parametrul special „$ @” ia întreaga listă și o separă în argumente separate

```
array[index]=value
```

```
VAR_ARRAY[0]=1
```

```
VAR_ARRAY[1]=3
```

Pentru accesarea valorii:

```
${array[index]}
```

```
${array[*]} sau ${array[@]}
```

SHELL Programming - Operatori

Sunt suportate mai multe tipuri de operatori:

- Aritmetici: +, -, *, /, %, =, ==, != `**expr** $a + $b`

  expr este un program extern; Trebuie sa existe spațiu între operatori și expresii; expresia completa trebuie incadrata între ` ` (backtick)

- Relationali: (numere) -eq, -ne, -gt, -lt, -ge, -le
- Booleeni: !, -o (OR), -a (AND)
- Stringuri: = (egalitate), !=, -z (zero size), -n (non-zero size), str (empty)
- Test pe fisiere: -d, -f, -r, -w, -x, -s (size>0), -e (exists, fisier sau folder), -p, -b, -c,

SHELL Programming - Exercises

Numărați toate liniile de cod din fișierele C din directorul dat ca argument al liniei de comandă, excluzând liniile care sunt goale sau conțin doar spații goale:

```
#!/bin/bash

S=0
for f in $1/*.c; do
        N=`grep "[^ \t]" $f | wc -l`
        S=`expr $S + $N`
done
echo $S
```

!!! Atentie la numele de fisier care contin spatiu

SHELL Programming

Filename wildcards: similare dar mai simple decat expresiile regulate

Reguli:

* - orice secvență de caractere, inclusiv secvența vida, dar nu primul punct din numele de fișier

? - orice caracter (1 singur), dar nu primul punct din numele de fișier

[abc] – Listă de opțiuni de caractere, suportă rangeuri precum in expresiile regulate

[!abc]- Negarea listă de opțiuni de caractere (similar cu [^abc] din regex)

Exemplu: lista fișierelor a căror nume începe cu o literă și au o extensie de exact două caractere ls [a-zA-Z]*.??

SHELL Programming

Care este rezultatul execuției următorului cod:

```
$ cd /

$ ls -ld {,usr,usr/local}/{bin,sbin,lib}
```

SHELL Programming - Decision

- if - fi

```
if [ expression ]
then
   Statements to be executed when
expression is true
fi
```

- if -else - fi

```
if [ expression ]
then
   Statements to be executed when
expression is true
else
   Statements to be executed when
expression is false
fi
```

- if - elif - else - fi

```
if [ expression 1 ]
then
   Statements to be executed if expression 1 is
true
elif [ expression 2 ]
then
   Statements to be executed if expression 2 is
true
else
   Statements to be executed if no expression
is true
fi
```

SHELL Programming - Decision

Similar to multiple if - elif statements

```
case word in
   pattern1)
                Statements to be executed if pattern1
matches
                ;;
   pattern2)
                Statements to be executed if pattern2
matches
                ;;
   *)
                Default condition to be executed
                ;;
esac
```

SHELL Programming - Loops

- while loop: execută comenzile date până când rămâne condiția dată true
- for loop
- until loop: se execută până când condiția devine adevărată
- select loop

- break: termină execuția întregii bucle
- continue: determină ieșirea din bucla curentă dar nu din întreaga buclă

break n **continue n**

SHELL Programming - Exercitii

Numărați toate liniile de cod din fișierele C din directorul dat ca argument al liniei de comandă și subdirectoarele sale, exclusiv liniile care sunt goale sau conțin doar spații goale

```
#!/bin/bash

S=0
for f in `find $1 –type f –name “*.c”; do
        N=`grep "[^ \t]" $f | wc -l`
        S=`expr $S + $N`
done
echo $S
```

SHELL Programming - Conditions

Ce inseamna/face **test**?

Pentru ca scrierea condiției să pară ceva mai naturală, există o a doua sintaxă, în care [este un alias al comenzii pentru test și] marchează sfârșitul testului. Atenție, lăsați spații în jurul acestor paranteze pătrate sau vor exista erori de sintaxă.
Exemplul de bază IF din prezentare poate fi rescris după cum urmează:

```
for A in $@; do
        if [ -f $A ]; then
        echo $A is a file
        elif [ -d $A ]
        then
        echo $A is a dir
        elif echo $A | grep -q "^[0-9]\+$";
then
        echo $A is a number
        else
        echo We do not know what $A is
        fi
done
```

SHELL Programming - Exercitii

Citiți intrarea consolei până când utilizatorul furnizează un nume de fișier care există și poate fi citit

```
#!/bin/bash

F=””
while [ -z "$F" ] || [ ! -f "$F" ] || [ ! -r "$F" ]; do
                read -p "Provide an existing and readable file path:" F
done
```

```
#!/bin/bash

F=””
while test -z "$F" || ! test -f "$F" || ! test -r "$F"; do
                read -p "Provide an existing and readable file path:" F
done
```

SHELL Programming - Exercitii

Scrieți un script care monitorizează starea unui director și tipărește o notificare atunci când ceva s-a schimbat

```
#!/bin/bash
D=$1
if [ -z "$D" ]; then
  echo "ERR: No directory for monitoring" >&2
  exit 1
fi
if [ ! -d "$D" ]; then
  echo "ERROR: Directory $D does not exist" >&2
  exit 1
fi
STATE=""
while true; do
                S=""
                for P in `find $D`; do
                if [ -f $P ]; then
                LS=`ls -l $P | sha1sum`
                CONTENT=`sha1sum $P`
                else
                LS=`ls -l –d $P | sha1sum`
                CONTENT =`ls -l $P | sha1sum`
                fi
                S="$S\n$LS $CONTENT"
                done
                if [ -n "$STATE" ] && [ "$S" !=
"$STATE" ]; then
                echo "Directory state changed"
                fi
                STATE=$S
                sleep 1
```