Laborator 5

Shell

Shell - un program interactiv care preia comenzi de la utilizator și le trimite către sistemul de operare pentru a fi rulate.

Shell-ul este un program special conceput pentru a fi o interfață între utilizator și sistemul de operare.

Istoric: RUNCOM (1964), Louis Pouzin -> Multics Shell (1965) , Glenda Schroeder -> Thompson Shell (1971), Ken Thompson -> Bourne Shell (1979), Stephen Bourne -> BASH - Bourne Again SHell (1989) Brian Fox

Pe orice sistem Linux se pot instala mai multe shell-uri (dash, csh, ksh, bash)
Windows are cmd, PowerShell

`cat /etc/shells` - afișează toate programele instalate în sistem care au posibilitatea de a fi shell-uri. Dar fișierul /etc/shells trebuie analizat cu atenție, în el apar ocazional și programe care nu sunt shell (ex: screen)

Shell-urile sunt interpretoare de comenzi, au posibilitati de programare

`echo $SHELL` - afișează shell-ul default al utilizatorului

`ps -p $$` - afișează shell-ul care rulează în momentul respectiv

Editoare de text

vi

http://www.atmos.albany.edu/daes/atmclasses/atm350/vi_cheat_sheet.pdf

```
cd ~
vi fisiertest
```

Vi are doua moduri: modul de comanda (care permite salvarea fișierului, copy, paste si multe altele) si modul de scriere. Vi incepe intotdeauna in modul de comanda. Intrarea in modul de scriere se face apasand una dintre comenzile de input (cele mai frecvente sunt a sau i).

Intrati in modul editor, scrieti un text
Iesiti din modul editor (esc), salvați fișierul apasand :w
Intrati in modul editor, scrieți altceva
Iesiti din modul editor (esc), apasati :q - nu merge
Apasati :wq (write + quit)

```
vi fisiertest
```

Intrati in modul editor, scrieti ceva

Iesiti din modul editor (esc) apasati :q!
Editorul iese fără sa salveze

Dacă intrăm în vi fără nici un nume de fișier pentru a salva trebuie specificat numele fișierului la comanda :w- :w file.txt sau :wq - :wq file.txt

nano

https://www.nano-editor.org/dist/latest/cheatsheet.html

mcedit

https://cheatography.com/hank/cheat-sheets/midnight-commander/

Variabile de mediu

Orice proces in Linux poate avea o serie de informații stocate ca variabile de mediu.
La pornirea oricărui shell acesta va prelua o serie de variabile de mediu pe care le va transfera programelor pe care le porneste prin fork

`cat /proc/$$/environ` - lista variabilelor de mediu care sunt disponibile interpretorului curent
`printenv` - comanda care afișează același lucru

Nu toate procesele au acces la aceeași lista de variabile de mediu.

Ex: dacă exista un proces crond pe sistem (de regula exista):

```
sudo cat /proc/$(pidof crond)/environ
```

comparati cu

```
cat /proc/$$/environ
```

Multe erori in programele (scripturile) shell vin din necunoașterea variabilelor de mediu care vor fi disponibile la rularea programului.

Input/Output

read, echo

```
#!/bin/bash

echo -n "Introduceti ceva:"
```

```
read sir

echo "Ati introdus: $sir"
```

printf

```
#!/bin/bash

sd=12
mf=35

printf "ala bala %d %d\n" $sd $mf
```

Hello World

Simplu

Creati un fisier numit: hellow cu următorul conținut:

```
#!/bin/bash
echo "Hello World"
```

Transformați fișierul într-un fișier executabil

chmod +x hellow

rulati fișierul ca pe un program obișnuit:

./hellow

Hello world cu variabile

```
#!/bin/bash

var="Hello World"
echo "$var"
```

Variabile

Definirea variabilelor

Limbajul BASH este netipizat - toate variabilele sunt tratate ca și cum ar fi șiruri de caractere

```
#!/bin/bash

numar=13
echo "$numar"
```

Variabilele care conțin spații trebuie puse între ghilimele

```
#!/bin/bash

text=”ala bala”
echo "$text"
```

Variabilele se pot prelua din comenzi

```
#!/bin/bash

NOW=$(date +"%m-%d-%Y")
echo $NOW
```

Vectori

Operații cu vectori:

arr=()	Creaza un vector gol
arr=(1 2 3)	Initializeaza un vector
${arr[2]}	Al treilea element
${arr[@]}	Toate elementele
${!arr[@]}	Toti indicii
${#arr[@]}	Dimensiunea vectorului

arr[0]=3	Înlocuiește primul element
arr+=(4)	Adaugă o valoare la vector
arr=($(ls))	Vector de fișiere preluat din comanda ls
${arr[@]:s:n}	Elementele vectorului care pornesc de la indexul s

```
#!/bin/bash

arr=( $(ls /) )
arrdim=${#arr[@]}
arrthree=${arr[2]}

echo "Al teilea $arrthree"
echo "Dimensiune: $arrdim"
echo ${arr[@]}
```

Operatiile aritmetice

expr

expr este o comanda disponibilă în sistemul de operare care evalueaza expresii aritmetice:

expr se poate rula în afara scriptului:

expr 3 + 4 (7)
expr 3 * 4 (12)
expr 6 / 3 (2)
expr 6 / 4 (1) - expr operează cu numere întregi

```
#!/bin/bash
x=5
y=10
ans=$(expr $x + $)
echo "$ans"
```

utilizarea instrucțiunii let

let este o instrucțiune a bash-ului care permite evaluarea instrucțiunilor aritmetice

```
#!/bin/bash

x=5
y=10
let ans=$x+$y
echo "$ans"
```

expansiune aritmetica

```
#!/bin/bash
x=5
y=10
ans=$(( x + y ))
echo "$ans"
```

Structuri decizionale

if, then, elif, fi

```
#!/bin/bash

NAME="Bill"
if [ "$NAME" = "John" ]; then
  echo "True - my name is indeed John"
elif [ "$NAME" = "Bill" ]; then
  echo "Am gasit un Bill";
else
  echo "False"
fi
```

Comparațiile valorilor numerice

Operatie	Se evaluează ca adevărată dacă:
$a -lt $b	$a < $b
$a -gt $b	$a > $b
$a -le $b	$a <= $b

$a -ge $b	$a >= $b
$a -eq $b	$a nu este egal cu $b
$a -ne $b	$a nu este egal $b

Comparațiile șirurilor

Operație	Se evaluează ca adevărată dacă:
"$a" = "$b"	$a este acelasi ca $b
"$a" == "$b"	$a este acelasi ca $b
"$a" != "$b"	$a este diferit de $b
-z "$a"	$a este gol

Operatori logici

&& AND
|| OR
! NOT

Case / Esac

```
#!/bin/bash

NAME="Marcel"
echo -n "Mancarea preferata a lui $COUNTRY este "

case $NAME in

  Marcel)
    echo -n "Lasagna"
    ;;

  Ionel | Vasile)
    echo -n "Salata"
    ;;

Gheorghe | Vlad | Radu)
    echo -n "Tofu"
    ;;

  *)
```

```
    echo -n "Nu am NICI O IDEE"
    ;;
esac
```

Structuri repetitive

while

```
#!/bin/bash

counter=1

while [ $counter -le 10 ]
do
        echo $counter
        ((counter++))
done
echo All done
```

until

```
#!/bin/bash

counter=1

until [ $counter -gt 10 ]

do
        echo $counter
        ((counter++))
done
echo All done
```

for

```
#!/bin/bash
```

```
names='Stan Kyle Cartman'

for name in $names
do
        echo $name
done

echo All done
```

intervale

```
for value in {1..5}
for value in {10..0..2}
```

ciclare prin vector

```
#!/bin/bash

arr=( $(ls /) )
arrdim=${#arr[@]}
#arrthree=${arr[2]}

#echo "Al teilea $arrthree"
echo "Dimensiune: $arrdim"
#echo $arr

for file in ${arr[@]}
do
       echo $file
done
```

select

```
names='Kyle Cartman Stan Quit'

PS3='Select character: '

select name in $names
do

if [ $name == 'Quit' ]
```

```
        then
        break
fi
echo Hello $name
done
echo Bye
```

Exemple

https://github.com/NARKOZ/hacker-scripts