Laboratorul 7

1. Scrierea scripturilor sub Linux

Pe lângă execuţia comenzilor citite de la un terminal, shell-ul poate controla şi execuţia unor comenzi citite dintr-un fişier. Astfel de fişiere se numesc **scripturi** sau **proceduri shell** (“shell scripts”). Pe lângă comenzile Linux propriu-zise, aceste fişiere pot conţine structuri pentru controlul fluxului. Scripturile se pot edita ca orice fişier text.

Pentru a se indica shel-ului dorit să interpreteze comenzile din scriptul respectiv, prima linie din script este de forma: `#!/bin/nume_shell`

De obicei, caracterul # indică începutul unui comentariu; face excepţie situaţia în care al doilea caracter este `!`, caz în care shell-ul interpretează restul liniei ca şi o comandă shell pe care o execută. Dacă se foloseşte shel-ul implicit (`bash`) această linie nu mai este necesară.

După salvarea fişierului, pentru ca numele său să poate fi folosit ca orice altă comandă Linux, trebuie setat dreptul său de execuţie, folosind o comandă `chmod.`

Exemplu. Dacă `nume_script` este numele cu care a fost salvat scriptul respectiv şi suntem în directorul în care a fost salvat scriptul, dacă se tastează:

```
$chmod a+x nume_script
```

Se va seta dreptul de execuţie pentru toţi utilizatorii.

Pentru lansarea în execuţie, se tastează o linie de forma:

```
$ cale\nume_script [Lista_arg]
```

`cale` indică directorul unde a fost salvat scriptul respectiv;

`Lista_arg` este de forma `arg1 arg2 ... argn`.

Orice script poate fi parametrizat prin utilizarea parametrilor formali, notaţi cu `$0`,..., `$n` , care pot fi utilizaţi ca denumiri ale unor entităţi din interiorul scriptului. Asocierea argumentelor din linia de comandă cu parametrii formali dintr-un script shell se face astfel: `$0` se substituie cu numele scriptului (comanda), în timp ce `$i` se substituie cu al `i`-lea argument din linia de comandă (`i = 1,..,9`). Dacă sunt mai puţin de `9` argumente, cele absente au ca valoare şirul vid. Dacă comanda este lansată cu mai mult de `9` argumente, se vor reţine toate, dar la un moment dat vor putea fi referite doar primele `9` prin intermediul construcţiilor de forma `$1` până la `$9`, plus numele comenzii reţinut în `$0`.

2. Controlul secvenţial

`cda1 && cda2` are semnificaţia:

- se execută `cda1`;
- dacă execuţia acesteia s-a terminat cu succes (cod de retur zero), se execută şi `cda2`.

`cda1 || cda2` are semnificaţia:

- se execută `cda1`;
- dacă execuţia acesteia se termină cu eşec (cod de retur nenul) se execută şi `cda2`.

Exemplu. Dacă se tastează:

```
$cd /home/ilflorea && ls -l
```

Se afişează conţinutul directorului `/home/ilflorea`, numai dacă acesta există (comanda `cd` s-a terminat cu succes).

Exemplu. Dacă se tastează:

```
$cd /home/ilflorea && echo Director inexistent
```

Se afişează mesajul `Director inexistent` numai dacă directorului `/home/ilflorea` nu există (comanda `cd` s-a terminat cu eşec).

Caracterul `;` separă două comenzi care se vor executa succesiv. Delimitatorul de comenzi permite scrierea mai multor comenzi pe aceeaşi linie. Trecerea la execuţia comenzii următoare se face numai după terminarea execuţiei comenzii precedente din aceeaşi linie.

Astfel, într-o linie de forma: `cmd1; cmd2; cmd3`
execuţia comenzii `cmd2` începe numai după terminarea lui `cmd1`, iar a lui `cmd3` numai după terminarea lui cmd2.

Exemplu.
```
$pwd;ls;cd ../ilflorea;pwd;ls >/tmp/iesiri.out
```
Redirectarea finală se aplică numai ultimei comenzi `ls`. Ieşirile celorlalte comenzi apar pe ecran (nu sunt redirectate).

Observaţie. Trebuie făcută deosebirea între această construcţie şi legarea în *pipe*; prin legare în *pipe*, pot fi lansate simultan în execuţie, în procese separate, mai multe comenzi.

Gruparea comenzilor. O succesiune de comenzi delimitată de paranteze sau de acolade şi separate prin `;` contează sintactic ca o singură comandă. Efectul execuţiei unei astfel de comenzi este diferit în funcţie de delimitatorii folosiţi. Astfel, comenzile dintre paranteze nu afectează mediul exterior, pe când cele dintre acolade modifică acest mediu.

Exemplu.
```
${ pwd;ls;cd ../ ilflorea;pwd;ls; } >/tmp/iesiri.out
```
Redirectarea se va aplica ieşirilor tuturor comenzilor grupate.
Pentru:
```
$( pwd;ls;cd ../ ilflorea; pwd; ls ) >/tmp/iesiri.out
```
Aplicarea redirectării va afecta numai interiorul grupului.

Exemplu.
```
$( pwd;ls |grep fis) | wc
```
are ca efect faptul că pentru comanda wc fişierul de intrare este constituit din: fişierul de ieşire produs de `pwd` la care se adaugă rezultatul legării în pipe `ls| grep`.
Aceeaşi efect se obţine prin:
```
$pwd >/tmp/tmp$$; ls | grep >>/tmp/tmp$$
$wc /tmp/tmp$$
```

Exemplu. Presupunem că directorul curent este `/home/ilflorea`. Execuţia comenzii:
```
$ pwd; (cd ..; pwd;); pwd
```
are ca efect afişarea următoarelor linii:
```
/home/ilflorea
/home
/home/ ilflorea
```
În schimb, comanda:
```
$pwd; { cd ..; pwd; }; pwd
```
va afişa:
```
/home/ilflorea
/home
/home
```
În cazul grupării comenzilor `cd` şi `pwd` între paranteze rotunde, acestea au efect doar în interiorul lor, în timp ce în cazul grupării comenzilor între acolade, rezultatul execuţiei lor este global, fiind vizibil şi în exteriorul grupului.

3. Condiţii sub Linux

Linux recunoaşte următoarele condiţii elementare:
a) comparaţii numerice;
b) teste asupra şirurilor de caractere;
c) teste asupra fişierelor.

Aceste condiţii elementare pot fi legate între ele prin operatorii:

-a joacă rolul de SI logic (AND);
-o joacă rolul de SAU logic (OR);
! este operatorul unar de negare (NOT).

Pentru gruparea unor subexpresii se folosesc construcţii de forma `( ... )`. Deoarece parantezele sunt caractere speciale, ele trebuie scrise sub una din formele: `" ( " , ' ( ' , \` pentru paranteze deschise, respectiv formele `" ) " , ' ) ' , \ )` pentru parantezele închise.

Comparaţii numerice. Două expresii numerice pot fi comparate folosind operatorii relaţionali: `-lt -le -eq -ne -ge -gt,` care corespund operatorilor relaţionali din C: `< <= == != >= >`

Testele asupra şirurilor de caractere `sunt:`

`-z şir` verifică dacă şirul are lungimea zero;
`-n şir` verifică dacă şirul are lungime nenulă;
`s1 = s2` verifică dacă cele două şiruri sunt egale (unele sisteme de operare acceptă şi construcţia `==`);
`s1 != s2` verifică dacă cele două şiruri sunt diferite.

Testele asupra fişierelor sunt`:`

`-e fis` verifică dacă fişierul `fis` există;
`-s fis` verifică dacă fişierul `fis` există şi are lungimea nenulă;
`-r fis` verifică dacă fişierul `fis` există şi din el se poate citi;
`-w fis` verifică dacă fişierul `fis` există şi în el se poate scrie;
`-x fis` verifică dacă fişierul `fis` există şi este executabil;
`-f fis` verifică dacă fişierul `fis` există şi este un fişier obişnuit;
`-d fis` verifică dacă fişierul `fis` există şi este un director;
`-L fis` verifică dacă fişierul `fis` există şi este o legătură simbolică;
`-p fis` verifică dacă fişierul `fis` există şi este un pipe;
`-c fis` verifică dacă fişierul `fis` există şi este un fişier special de tip caracter;
`-b fis` verifică dacă fişierul `fis` există şi este un fişier special de tip bloc.

Testarea unei condiţii se face cu comanda `test conditie` sau `[ conditie ]` Dupa executia unei comnenzi, se trimite un *cod de retur* (sau *cod de eroare*). De obicei acesta este zero dacă execuţia comenzii s-a încheiat normal şi are o valoare nenulă în caz contrar. Comanda `test` primeşte ca argument o condişie pe care o evaluează. În funcţie de rezultatul acestei evaluări, se fixează valoarea codului de retur. În cazul în care condiţia este adevărată, codul de retur este fixat la valoarea `zero`, altfel (condiţie falsă) codul de retur este fixat la o valoare nenulă. Comanda este utilizata împreună cu structurile de control shell.

4. Structura alternativă `if`

Structura alternativă **`if`** are o sintaxă de forma:

```
if listaCom
          then listaCom
          [elif listaCom
               then listaCom
     --------------------
          elif listaCom
               then listaCom]
          [else listaCom]
fi
```

Lista de comenzi care urmează după `if`, ca şi listele de comenzi care urmează după `elif` au un dublu rol: de *execuţie* a comenzilor din listă şi de *fixare a valorii de adevăr a execuţiei*. O execuţie *are valoarea TRUE* dacă *codul de retur* al ultimei execuţii din lista de comenzi are *valoarea zero*. Execuţia are valoarea *FALSE* dacă codul de retur are *valoare nenulă*. Listele de comenzi de după `then`, ca şi lista de comenzi de după `else` au doar valori de execuţie.

Succesiunea de evenimente care au loc la întâlnirea unei comenzi `if` este următoarea:

- Se execută **lista de comenzi** ce urmează după `if`. Dacă rezulta valoarea `TRUE` (condiţia `if` este adevărată), atunci se execută lista de comenzi de după `then` şi execuţia lui `if` se termină (se trece la instrucţiunea care urmează după `fi`). În caz contrar (lista de comenzi de după `if` generează `FALSE`) se trece la pasul următor.
- Dacă există (una sau mai multe) construcţii `elif`, atunci se execută, pe rând, listele de comenzi care urmează după `elif`, până când una dintre ele generează valoarea `TRUE`. Apoi se execută lista de comenzi de după `then` - ul corespunzător şi execuţia lui `if` se termină. În caz contrar (fie nu există `elif`, fie toate listele de comenzi de după `elif` - uri au generat `FALSE`) se execută pasul următor.
- Dacă există `else` atunci se execută lista de comenzi de după `else` şi execuţia lui `if` se termină. În caz contrar (nu există `else`), execuţia lui if se termină şi se trece la execuţia comenzii ce urmează după `fi`.

Exemplu. Se testeaza daca un fişier este ordinar sau director.

```
if [ -f $1 ]
     then echo “$1 este un fişier ordinar”
     elif [ -d $1 ] then echo “$1 este un director”
     else echo “$1 este necunoscut”
fi
```

Exemplu. Scriptul urmator:

```
if grep “Ionel" fis_lista > /dev/null
     then echo "Numele a fost gasit in lista "
     else echo "Numele nu este in lista "
fi
```

caută numele `Ionel` in fisierul `fis_lista` şi afisează un mesaj în care se precizează rezultatul cautarii. Se foloseşte redirectarea iesirii (`>/dev/null`), pentru ca nu ne intereseaza liniile găsite, ci doar dacă există asemenea linii. În acest exemplu `if` testează o comandă.

Exemplu. Vom rescrie scriptul precedent, şablonul şi fişierul în care căutam sunt date ca argumente în linia de comandă.

```
if grep "$1" "$2" >/dev/null
     then echo "$1 apare in fisierul $2"
     else echo "$1 nu apare in fisierul $2"
fi
```

Exemplu. Rescriem scriptul precedent, testând corectitudinea liniei de comandă (se verifică dacă numărul argumentelor din linia de comanda este corect).

```
if [ $# -lt 2 ]
     then
     echo "Prea putini parametri"
     exit 1
fi
```

```
if grep "$1" "$2" >/dev/null 2>&1
     then echo "$1 apare in fisierul $2"
     else echo "$1 nu apare in fisierul $2"
fi
```
exit 0

Exemplu. Testarea de către `if` a altor comenzi.

```
# If testeaza comanda de comparare a doua fisiere
if cmp a b &> /dev/null # anularea iesirii
     then echo "Fisierele a si b sunt identice"
     else echo " Fisierele a si b sunt diferite"
fi
# If testeaza comanda de cautare intr-un fisier.
if grep -q Bash file
   then echo "Fis. contine cel put. o ap. a lui
Bash."
fi
# If testeaza legarea in pipe a doua comenzi
word=Linux
Secv_Lit=inu
if echo "$word" | grep -q "$Secv_Lit"
     then echo "$Secv_Lit gasita in $word"
     else echo "$Secv_Lit negasita in $word "
fi
```

Exemplu. Un fişier de comenzi care afişează liniile ordonate alfabetic ale unui fişier text. Numele fişierului va fi dat ca prim argument al linei de comandă. O primă variantă este:

```
if [ $# -eq 0 ]
  then echo "Trebuie dat un nume de fisier"
  else sort $1 | more
fi
```

Varianta următoare testează şi tipul fişierului.

```
if [ $# -eq 0 ]
then echo "Trebuie dat un nume de fisier"
elif [ ! \( -r $1 \) ]
then echo "Fisierul $1 nu exista"
else sort $1 | more
fi
```

Exemple. Utilizarea codului de retur al comenzii anterioare (conţinut în variabila `$?`).

```
$grep -q cuvant fisier
$if [ $? -eq 0 ];then echo Da;else echo Nu;fi
```

În comanda grep se foloseşte opţiunea `-q`, prin care nu se afişează rezultatul căutării. Secventa anterioară are acelaşi efect cu:

```
$ grep cuvant fisier >/dev/null
$ if [ $? -eq 0 ];then echo Da;else echo Nu;fi
```

În exemplul următor, se face ştergerea numai dacă comanda `cd` s-a executat cu succes.

```
cd mytmp
if (( $? )); then rm * ; fi
```

Se obţine acelaşi lucru dacă se lansează:

```
$cd mytmp && rm *
```

Exemplu. Un fişier de comenzi care compară directorul curent cu directorul dat ca parametru în linia de comandă.

```
dirc=`ls –l`
dirp=`ls -l $1`
if [ "$dirc" = "$dirp" ]
     then echo directoarele `pwd` si $1 coincid
     else echo directoarele `pwd` si $1 sunt dif.
fi
```

Cele două variabile (`dirc, dirp`) primesc ca valoare un şir de caractere, reprezentând rezumatul unui director (comanda `ls`). Când se face comparaţia (condiţia din `if`), este necesară prezenţa ghilimelelor, pentru ca fiecare variabilă să fie un singur şir de caractere. Dacă se rulează exemplul fără ghilimele, rezultatul va fi se afişează un mesaj de eroare.

5. Structura alternativă `case`

Structura alternativă `case` are sintaxa:

```
case $var in
 list_valori1 ) actiune1;;
 list_valori2 ) actiune2;;
                ...
  * ) actiune_implicita;;
 esac
```

Valoarea variabilei `var` (*discriminant* pentru `case`) se compară pe rând cu valorile specificate în `list_valori1`, `list_valori2` etc. În momentul identificării unei corespondenţe, se execută acţiunea asociată şi execuţia lui `case` se termină. Dacă valoarea variabilei nu corespunde cu nici o valoare din liste, se execută acţiunea asociată lui * (acţiunea implicită), care se află pe ultima poziţie. Valorile specificate într-o listă de valori se separă prin `|` şi pot fi exprimate şi ca expresii regulate.

Exemplu. Testarea existenţei de fişiere şi adăugare la sfârşitul unui fişier.

```
case $# in
1) if [ -w $1] then cat >>$1
   fi ;;
2) if [ \( -w $2 \) -a \( -r $2 \)]
                                                    t
   hen cat >>$2 <$1
   ;;
*) echo Fisier inexistent;;
Esac
```

Dacă există un singur parametru, conţinutul fişierului standard de intrare se adaugă la sfârşitul fişierului al cărui nume este dat prin acest parametru. Pentru doi parametri în linia de comandă, conţinutul fişierului indicat de primul parametru este adăugat la sfârşitul fişierului indicat prin al doilea parametru. Pentru alte situaţii se emite un mesaj de utilizare.

Exemplu. O generalizare a comenzii `cal`. Comanda cal primeşte ca parametri anul sau luna în formă numerică şi afişează calendarul lunii/anului respectiv/e. În cele ce urmează, este prezentat un script care extinde comanda `cal`, în sensul că argumentele comenzii pot fi date şi în alte formate. Dacă numele cu care este salvat scriptul este `calendar`, poate fi lansat:

```
$./calendar ianuarie 2010
$./calendar febr
$./calendar Decem 2010
```

Conţinutul scriptului este:

```
case $# in
0) set `date`; m=$2;y=$6 ;; #cazul fara arg.
1) m=$1;set `date`;y =$6 ;;#luna este sing. arg.
2) m=$1; y =$2 ;; #se specif. luna si anul
esac
Case $m in
  Ian*|ian*|Jan*|jan* ) m=1 ;;
  Feb*|feb*|Fev*|fev* ) m=2 ;;
  Mar*|mar* m=3 ;;
  Apr*|apr*|Avr*|avr* ) m=4 ;;
  Mai|mai ) m=5 ;;
  Iun*|iun*|Jun*|jun* ) m=6 ;;
  Iul*|iul*|Jul*|jul* ) m=7 ;;
  Aug*|aug*|Avg*|avg* ) m=8 ;;
  Sep*|sep*|Sec*|sec* ) m=9 ;;
  Oct*|oct* ) m=10 ;;
  Noi*|noi*|Nov*|nov* ) m=11 ;;
  Dec*|dec* ) m=12 ;;
  [1-9]|10|11|12) ;; #Luna se specif. num
  *) m=“” ;; anul se specif. explicit.
esac
/usr/bin/cal $m $y #apel cal
```

1. Scrieţi o comandă `if` care să testeze că numărul argumentelor din linia de comandă este 3, că primele două argumente sunt fişiere şi în caz afirmativ să se concateneze primele două fişiere în al treilea fişier.
2. Scrieţi un fişier de comenzi care compară directorul curent cu două directoare date ca parametri în linia de comandă.
3. Scrieţi o comandă `if` care să testeze o comandă `ls -l` aplicată primului argument dat în linia de comandă, cu ieşire în al doilea argument şi în caz de execuţie cu succes a lui `ls`, să caute în ieşirea comenzii al treilea argument dat în linia de comandă.
4. Scrieţi o comandă `case` care să testeze că argumentul(ele) date sunt director(are); dacă se dă un singur argument să se listeze conţinutul acestuia iar dacă se dau două argumente, primul să se copieze recursiv în al doilea şi să se afişeze noua structură a acestuia. Pentru alte numere de argumente date, să se afişeze un mesaj.