Λειτουργικά Συστήματα Ι
Ασκήσεις Πράξης

Τμήμα Μηχανικών Πληροφορικής & Υπολογιστών

Σχολή Μηχανικών

Σενάρια Φλοιού (shell scripts)

Δομές ελέγχου

- Ο φλοιός bash, όπως άλλωστε όλοι οι φλοιοί διαθέτουν δομές επιλογής (**if**, **case**) που μας επιτρέπουν να εκτελούμε εντολές εφόσον ισχύει ή όχι κάποια συνθήκη και δομές επανάληψης (**while**, **until**, **for**) με τις οποίες μπορούμε να επαναλαμβάνουμε εντολές μέχρι να επιτευχθεί μια συγκεκριμένη κατάσταση.
- Πριν δούμε αναλυτικά τις δομές ελέγχου θα πρέπει να αναλύσουμε την εντολή **test**, με την οποία μπορούμε να κάνουμε πλήθος ελέγχων.

test

Επιλογές	Η συνθήκη είναι αληθής αν
test STRING1 = STRING2	Τα αλφαριθμητικά είναι ίσα
test STRING1 != STRING2	Τα αλφαριθμητικά δεν είναι ίσα
-z STRING	Το αλφαριθμητικό είναι κενό
-n STRING	Το αλφαριθμητικό δεν είναι κενό
test INTEGER1 -eq INTEGER2	Οι ακέραιοι είναι ίσοι
test INTEGER1 -ne INTEGER2	Οι ακέραιοι δεν είναι ίσοι
test INTEGER1 -gt INTEGER2	Ο ακέραιος INTEGER1 είναι μεγαλύτερος από τον ακέραιο INTEGER2
test INTEGER1 -ge INTEGER2	Ο ακέραιος INTEGER1 είναι μεγαλύτερος ή ίσος από τον ακέραιο INTEGER2
test INTEGER1 -lt INTEGER2	Ο ακέραιος INTEGER1 είναι μικρότερος από τον ακέραιο INTEGER2
test INTEGER1 -le INTEGER2	Ο ακέραιος INTEGER1 είναι μικρότερος ή ίσος από τον ακέραιο INTEGER2

test

Επιλογές	Η συνθήκη είναι αληθής αν
test -f FILE	Το αρχείο FILE υπάρχει και είναι κανονικό αρχείο
test -d FILE	Το αρχείο FILE υπάρχει και είναι κατάλογος
test -b FILE	Το αρχείο FILE υπάρχει και είναι τύπου block
test -c FILE	Το αρχείο FILE υπάρχει και είναι τύπου character
test -r FILE	Το αρχείο FILE υπάρχει και παρέχεται το δικαίωμα ανάγνωσης

test

Στη γραμμή εντολών η κλήση της test ακολουθείται συνήθως από την εντολή **echo $?** για να ελέγξουμε την τιμή επιστροφής της. Αν η test επιστρέψει την τιμή 0 σημαίνει πώς η συνθήκη είναι αληθής ενώ αν επιστρέψει την τιμή 1 η συνθήκη είναι ψευδής.

```
$ test -f first.sh          # check if file exist
$ echo $?                   # exit status of last command
0                           # true
```

Η εντολή test μπορεί να χρησιμοποιηθεί με δύο τρόπους. Στη γραμμή εντολών χρησιμοποιείται με την πλήρη μορφή της π.χ. **test -f filename**, ενώ στα σενάρια χρησιμοποιείται συνήθως με τη συμπαγή της μορφή **[-f filename]**.

if

if συνθήκη-1
then
    εντολές-1
elif συνθήκη-2
then
    εντολές-2
…
else
   εντολές-ν
fi

islogged.sh

```
$ cat islogged.sh
#!/bin/bash
# islogged.sh – if/else
if who | grep $1 > /dev/null
then
    echo "$1 is logged in"
else
    echo "$1 is not logged in"
fi
```

islogged.sh

Ακολουθεί ένα παράδειγμα κλήσης του islogged.sh:

```
$ ./islogged.sh nemo          # check for user nemo
nemo is logged in
```

Αν καλέσουμε όμως το σενάριο χωρίς να δώσουμε όρισμα, η grep θα εμφανίσει ένα μήνυμα όπως το παρακάτω:

```
$ ./islogged.sh               # call without arguments
Usage: grep [OPTION]... PATTERN [FILE]...
Try 'grep --help' for more information.
 is not logged in
```

isloggedv2.sh

```
$ cat isloggedv2.sh
#!/bin/bash
if [ $# -eq 1 ]; then
   if who | grep $1 > /dev/null ; then
      echo "$1 is logged in"
   else
      echo "$1 is not logged in"
   fi
else
   echo Usage: $0 username
fi
$ ./isloggedv2.sh          # call without argument
Usage: ./isloggedv2.sh username
```

Λογικοί τελεστές

- Χρησιμοποιώντας λογικούς τελεστές προσθέτουμε μεγαλύτερη ευελιξία στα σενάριά μας. Οι λογικοί τελεστές είναι: **!** για τη λογική άρνηση (NOT), **&&** για τη λογική σύζευξη (AND) και **||** για τη λογική διάζευξη (OR).
- Οι τελεστές **-a** και **-o** της test είναι προς κατάργηση (deprecated) και συνιστάται να μη χρησιμοποιούνται πλέον.

call4pause.sh

```
$ cat call4pause.sh
#!/bin/bash
if [ $# -eq 2 ] && who | grep $1 1>/dev/null
then
  write $1 "Hi $1, let's go for $2?"
2>/dev/null
else
  echo No message was sent ...
fi

$ ./call4pause.sh Alice coffee
No message was sent ...
```

leapyear.sh

```
#!/bin/bash
echo -n "Give year: "; read year
if [ $(($year % 400)) -eq "0" ]; then
  echo "$year is a leap year."
elif [ $(($year % 4)) -eq "0" ]; then
  if [ $(($year % 100)) -ne "0" ]; then
    echo "$year is a leap year. "
  else
    echo "$year is not a leap year."
  fi
else
  echo "$year is not a leap year"
fi
```

case

case έκφραση **in**

πρότυπο-1**)** εντολές-1;;

πρότυπο-2**)** εντολές-2;;

…

πρότυπο-ν**)** εντολές-ν;;

esac

Η τιμή της έκφρασης συγκρίνεται διαδοχικά με τα πρότυπα μέχρι να βρεθεί ταίριασμα. Αν βρεθεί κάποιο ταίριασμα εκτελούνται οι αντίστοιχες εντολές μέχρι το διπλό ελληνικό ερωτηματικό. Στη συνέχεια τερματίζεται η case και εκτελείται η εντολή που την ακολουθεί.

case

```
#!/bin/bash
echo -n "Which Linux distribution do you use?"
read distro
case $distro in
  centos)
    echo "Very nice...";;
  ubuntu|debian)
    echo "It's my favorite OS!";;
  fedora)
    echo "I prefer Red Hat.";;
  *)
    echo "Never used it.";;
esac
```

case

- Από την έκδοση 4 και μετά του φλοιού μπορούμε να τερματίσουμε ένα μπλοκ εντολών εκτός από δύο ελληνικά ερωτηματικά **;;** και με τις ακολουθίες **;&** και **;;&**.
- Η ακολουθία ;& έχει ως συνέπεια να εκτελεστεί και το επόμενο μπλοκ εντολών χωρίς να γίνει έλεγχος για ταίριασμα του προτύπου ενώ με την ;;& γίνεται έλεγχος και στο επόμενο πρότυπο και εφόσον υπάρχει ταίριασμα εκτελούνται οι σχετικές εντολές του. Με αυτόν τον τρόπο μπορούμε να ελέγξουμε όλα τα πρότυπα ή να κάνουμε κοινή χρήση κώδικα.

levels.sh

```
#!/bin/bash
case "$1" in
  3)
    echo "Level Three";&
  2)
    echo "Level Two";&
  1)
    echo "Level One";&
esac
```

```
$ ./levels.sh 2
Level Two
Level One
```

for

Ο βρόχος for χρησιμοποιείται σε συνδυασμό με μία μεταβλητή η οποία παίρνει τιμές από ένα σύνολο προκαθορισμένων τιμών και εκτελεί τις εντολές του για κάθε μία τιμή. Η σύνταξη της εντολής έχει την παρακάτω μορφή:

for μεταβλητή **in** λίστα_τιμών

do

εντολές

done

hello2all.sh

```
#!/bin/bash
for name in Alice Bob Carol
do
   echo hello $name
done

$ ./hello2all.sh
hello Alice
hello Bob
hello Carol
```

hello2allv2.sh

Η λίστα τιμών συνήθως αποτελείται από έναν πίνακα, τα στοιχεία του οποίου η μεταβλητή λαμβάνει επαναληπτικά. Η έκδοση του προηγούμενου σεναρίου με χρήση ενός πίνακα έχει ως εξής:

```
#!/bin/bash
names=(Alice Bob Carol)        # declare -a names
for name in ${names[@]}
do
   echo hello $name
done
```

hello2allv3.sh

Ένας εναλλακτικός τρόπος χειρισμού ενός πίνακα είναι η προσπέλαση των στοιχείων του μέσω δείκτη:

```
#!/bin/bash
names=(Alice Bob Carol)
for i in ${!names[@]}                    # get indexes
do
  echo hello ${names[$i]}
done
```

random.sh

Από την έκδοση 2.04 και μετά ο φλοιός Bash υποστηρίζει μία έκδοση του βρόχου for παρόμοιου με αυτόν της γλώσσας προγραμματισμού C.

```
#!/bin/bash
for (( i=1; i<=6; i++ ))   # new C-style for
do
  printf "%d " $((RANDOM % 49 + 1))
done
echo
```

```
$ ./random.sh
4 9 36 46 22 15
```

while

Ο βρόχος while επαναλαμβάνει την εκτέλεση μιας ομάδας εντολών όσο ισχύει μία συνθήκη. Η σύνταξη της εντολής έχει την παρακάτω μορφή:

while συνθήκη

do

εντολές

done

myseq.sh

```
#!/bin/bash
if [ $# -ne 2 ]; then    # check for 2 arguments
  echo Usage: $0 start stop
  exit 1
fi
a=$1 ; b=$2
while [ $a -le $b ]
do
  printf "%d " $a
  (( a++ ))
done
echo
```

until

Ο βρόχος **until** μοιάζει με τον βρόχο while με τη διαφορά πώς οι εντολές του εκτελούνται μέχρι η συνθήκη του να γίνει αληθής. Όσο δηλαδή η συνθήκη είναι ψευδής εκτελούνται και οι εντολές. Η σύνταξη της εντολής έχει την παρακάτω μορφή:

until συνθήκη
do
    εντολές
done

sumofsquares.sh

```
#!/bin/bash
echo -n "Enter the number: " ; read n
i=1
sum=0
until [ $i -gt $n ]
do
    sq=$(( i * i))
    sum=$(( sum + sq ))
    (( i++ ))
done
printf "Sum of squares of first %d numbers is \
%d\n" $n $sum
```

sumofsquares.sh

Ο βρόχος until εκτελείται όσο η συνθήκη του δεν ισχύει, σταματάει δηλαδή να εκτελείται μόλις η μεταβλητή **i** ξεπεράσει τον αριθμό **n** που έδωσε ο χρήστης από το πληκτρολόγιο. Αρχικά υπολογίζεται το τετράγωνο της i, στη συνέχεια προστίθεται η τιμή αυτή στην μεταβλητή sum και τέλος αυξάνεται η i κατά 1. Ένα στιγμιότυπο εκτέλεσης του σεναρίου sumofsquares.sh έχει ως εξής:

```
$ ./sumofsquares.sh
Enter the number: 10
Sum of squares of first 10 numbers is 385
```

break - continue

Σε πολλές περιπτώσεις είναι επιθυμητό η πρόωρη εγκατάλειψη ενός βρόχου ή η παράλειψη κάποιων εντολών του. Η εγκατάλειψη ενός βρόχου γίνεται με την ενσωματωμένη εντολή **break** και η παράλειψη εντολών και έναρξη της επόμενης επανάληψης με την ενσωματωμένη εντολή **continue**. Προφανώς οι εντολές αυτές εκτελούνται μόνο εφόσον ικανοποιείται κάποια συνθήκη. Και οι δύο εντολές δέχονται έναν ακέραιο αριθμό **n** ως προαιρετικό όρισμα (**n ≥ 1)**, που καθορίζει το πλήθος των εμφωλευμένων βρόχων που θα εγκαταλείψει η break ή παραλείψει continue.

Συναρτήσεις - Ορισμός

Ο βασικότερος λόγος χρήσης συναρτήσεων είναι η επανα-χρησιμοποίηση του κώδικα. Είναι προφανές πώς ο ορισμός μιας συνάρτησης πρέπει να προηγείται της κλήσης της. Ο ορισμός μιας συνάρτηση φλοιού έχει την παρακάτω σύνταξη:

```
function όνομα_συνάρτησης()
{
   εντολές
}
```

Η δεσμευμένη λέξη function είναι προαιρετική. Αν όμως χρησιμοποιηθεί στον ορισμό τότε μπορούν να παραληφθούν οι παρενθέσεις. Ο κωδικός εξόδου μιας συνάρτησης είναι ο κωδικός εξόδου της τελευταίας εντολής του σώματός της.

Συναρτήσεις - Παράδειγμα

```
#!/bin/bash
# functions.sh
function sayHello()
{
  echo Hello
}

sayHelloToUser()
{
  echo Hello $1
}
```

```
sayHello
sayHelloToUser Alice
sayHelloToUser Bob
```

```
$ ./functions.sh
Hello
Hello Alice
Hello Bob
```

Συναρτήσεις - Καθολικές μεταβλητές

```
#!/bin/bash
# globalvar.sh

function myfunc()
{
  result= 'Docendo discimus'
}

myfunc
echo $result
```

```
$ ./globalvar.sh
Docendo discimus
```

Συναρτήσεις - Επιστροφή τιμής

```
#!/bin/bash
power()         # computes the power of a number
{
  local pow=1                       # local variable
  for (( i=1; i<=$2; i++ ))
  do
     (( pow *= $1 ))
  done
  echo $pow                         # return $pow
}
ans=$(power 5 3); echo 5^3=$ans    # power 5 3
```

Ερωτήσεις