Εισαγωγή στα Λειτουργικά Συστήματα

SET ΔΙΑΦΑΝΕΙΩΝ 11

SHELL VARIABLES & QUOTES

ΑΝΤΩΝΗΣ ΣΙΔΗΡΟΠΟΥΛΟΣ

Τι περιέχουν τα προγράμματα φλοιού

- Μεταβλητές
- Λογικές δομές (if, case …)
- Δομές επανάληψης (while, for, until)
- Συναρτήσεις (functions)
- Σχόλια
- Εκτέλεση προγραμμάτων/εντολών

Μεταβλητές

- Στο shell υπάρχουν μεταβλητές (εκτός ων μεταβλητών περιβάλλοντος) .
- Μπορούμε να ορίσουμε οποιαδήποτε μεταβλητή.
- Όλες οι μεταβλητές είναι τύπου “string” και δεν καθορίζουμε τον τύπο τους.

Μεταβλητές

- Ως ονόματα μεταβλητών ο χρήστης μπορεί να θέσει οποιονδήποτε συνδυασμό γραμμάτων και αριθμών (αρχίζοντας από γράμμα), ενώ από τους ειδικούς χαρακτήρες ο μόνος που μπορεί να χρησιμοποιείται με ασφάλεια είναι η υπογράμμιση “_”. Δεν επιτρέπονται κενά και άλλοι ειδικοί χαρακτήρες που μπορεί να έχουν κάποια ειδική σημασία για το κέλυφος (π.χ. $ \ # ; κ.ο.κ.).

Μεταβλητές κελύφους vs. μεταβλητές περιβάλλοντος

- Το sh χειρίζεται τις μεταβλητές περιβάλλοντος με τον ίδιο τρόπο που χειρίζεται τις μεταβλητές του shell.
- Συνηθίζεται για τις μεταβλητές περιβάλλοντος να χρησιμοποιούνται μόνο κεφαλαίοι χαρακτήρες ενώ για τις μεταβλητές του shell μόνο πεζοί.
- Οι μεταβλητές του shell αποθηκεύονται στην μνήμη που χρησιμοποιεί το ίδιο το shell, είναι τοπικές και δεν κληρονομούνται από τις διεργασίες στις θυγατρικές.
- Οι μεταβλητές περιβάλλοντος αποθηκεύονται από τον πυρήνα στο χώρο μνήμης που διατηρούνται τα διάφορα στοιχεία των διεργασιών και κληρονομούνται από τις διεργασίες στις θυγατρικές τους.

Μεταβλητές

- Ο ορισμός μιας μεταβλητής γίνεται με την αρχικοποίηση της:
- `a=5` (ορίζεται η μεταβλητή με όνομα α και αποθηκεύεται η τιμή 5 σαν string!!!!)
- `a="5"` (ισοδύναμο με το προηγούμενο)
- `a=" 5 "` (αποθηκεύεται η τιμή " 5 " σαν string - περιέχονται και τα κενά διαστήματα)
- **Προσοχή:** πριν και μετά τον χαρακτήρα "=" δεν πρέπει να υπάρχουν κενά!!!!
- **Προσοχή:** εάν στην τιμή που θέλουμε να εκχωρήσουμε περιέχονται ειδικοί χαρακτήρες (πχ: space * ; < > | κ.ο.κ.) τότε πρέπει να χρησιμοποιήσουμε εισαγωγικά (μονά ή διπλά) ή να αναιρέσουμε την ειδική σημασία αυτών των χαρακτήρων με το \.

Μεταβλητές

- τι λάθος συμβαίνει σε κάθε περίπτωση από τις παρακάτω;

```
asidirop@dellpc:~$ a= 5
5: command not found
asidirop@dellpc:~$ a= '5  '
5: command not found
asidirop@dellpc:~$ a =5
a: command not found
asidirop@dellpc:~$ a=5
asidirop@dellpc:~$
```

Μεταβλητές


```
asidirop@dellpc:~$ a =5
a: command not found
```

- το shell δεν "κατανοεί" την εκχώρηση. Νομίζει ότι προσπαθώ να εκτελέσω την εντολή με όνομα a και να της δώσω ένα όρισμα, το "=5".

Μεταβλητές


```
asidirop@dellpc:~$ a= 5
5: command not found
```

- το shell δεν "κατανοεί" την εκχώρηση.
- Νομίζει ότι προσπαθώ να εκτελέσω την εντολή με όνομα 5. Και πως ερμηνεύει το a= ?
- Υπάρχει η γενική σύνταξη:

```
ENV_VAR1=VAL1 ENV_VAR2=VAL2 .... command [args]
```

πχ:

```
A=563 B=jhfhds ls
```

Εκτελείται η εντολή ls αλλά στις μεταβλητές περιβάλλοντός της προστίθενται οι μεταβλητές περιβάλλοντος A και B

Μεταβλητές


```
asidirop@dellpc:/tmp$ env | grep '^LANG='
LANG=en_US.utf8
asidirop@dellpc:/tmp$ LANG=el_GR.UTF-8 env | grep '^LANG='
LANG=el_GR.UTF-8
asidirop@dellpc:/tmp$ env | grep '^LANG='
LANG=en_US.utf8
```

- Η env τυπώνει όλες τις μεταβλητές περιβάλλοντος. Με την grep ψάχνουμε μόνο την μεταβλητή περιβάλλοντος LANG.
  - στην 1η περίπτωση το LANG είναι en_US.utf8
  - στην 2η περίπτωση το LANG είναι el_GR.utf8 (για την εντολή env μόνο)
  - στην 3η περίπτωση βλέπουμε ότι η LANG από το τρέχον περιβάλλον έχει μείνει αμετάβλητη.

Ορισμός

- Ο ορισμός τιμής σε μια μεταβλητή μπορεί να γίνει με 2 τρόπους:
  - ανάθεση τιμής, πχ:
    - a=543245
    - b="$a"
  - ανάγνωση τιμής από την κανονική είσοδο χρησιμοποιώντας την read, πχ:
    - read a (θα ζητήσει να διαβάσει μια γραμμή από την κανονική είσοδο - προφανώς αν δεν ορίσουμε διαφορετικά θα την διαβάσει από το τερματικό - και θα εκχωρήσει την γραμμή (string) που διάβασε στην μεταβλητή του shell a.

2. Χρησιμοποιώντας την εντολή read

- Η εντολή read εκχωρεί σε μια μεταβλητή οτιδήποτε εισάγεται από το πληκτρολόγιο, ακολουθούμενο από μια αλλαγή γραμμής.

```
bash-2.05a$ read b
foo bar
bash-2.05a$ echo $b
foo bar
bash-2.05a$ echo "$b"
foo bar
bash-2.05a$
```

Χρησιμοποιώντας την εντολή read

- Η επιλογή -n στην echo δεν αλλάζει γραμμή μετά την εκτύπωση του μηνύματος. Συνηθίζεται όταν ζητούμε είσοδο δεδομένων από τον χρήστη.
- Η read θα διαβάσει μέχρι να πατήσουμε "αλλαγή γραμμής" (Enter).

read_demo

```
#!/bin/bash

echo -n "Enter some text > "
read text
echo "You entered: $text"
```

```
asidirop@aetos:/tmp$ ./read_demo
Enter some text > My name is Antonis
You entered: My name is Antonis
asidirop@aetos:/tmp$
```

Χρήση μεταβλητών

- Η χρήση μιας μεταβλητής (δηλαδή της τιμής που περιέχει) γίνεται χρησιμοποιώντας τον χαρακτήρα $.
- Ο $ "λέει" στο shell να θεωρήσει την λέξη (string) που ακολουθεί ως όνομα μεταβλητής και να χρησιμοποιήσει την τιμή της.

```
asidirop@dellpc:/tmp$ a=Hello
asidirop@dellpc:/tmp$ echo $a
Hello
asidirop@dellpc:/tmp$ echo ${a}zzz
Hellozzz
asidirop@dellpc:/tmp$ echo "${a}zzz"
Hellozzz
```

Τα εισαγωγικά

- Γενικώς τα εισαγωγικά αγνοούνται από το shell, και δεν περιλαμβάνονται τα ίδια στις τιμές των Strings

```
bash-2.05a$ echo "TEST"
TEST
bash-2.05a$ echo 'TEST'
TEST
bash-2.05a$ echo "TEST     A"
TEST     A
bash-2.05a$ echo TEST     A
TEST A
```

(Το shell αγνοεί τα πολλαπλά κενά αν αυτά δεν είναι μέσα σε εισαγωγικά)

Τα εισαγωγικά

- Καλό είναι όταν αναθέτουμε τιμή σε μια μεταβλητή ή όταν χρησιμοποιούμε μια μεταβλητή να χρησιμοποιούμε εισαγωγικά.
- Αν δεν χρησιμοποιούμε εισαγωγικά θα ερμηνευτούν οι ειδικοί χαρακτήρες. (το space είναι ειδικός χαρακτήρας)

```
bash-2.05a$ a='test    1'
bash-2.05a$ echo $a
test 1
bash-2.05a$ echo "$a"
test    1
bash-2.05a$
```

Τα εισαγωγικά

- Αν δεν χρησιμοποιούμε εισαγωγικά θα ερμηνευτούν οι ειδικοί χαρακτήρες. (το * είναι ειδικός χαρακτήρας)

```
asidirop@dellpc:/tmp$ a='*'
asidirop@dellpc:/tmp$ echo "$a"
*
asidirop@dellpc:/tmp$ echo $a
file1 file5 test8
asidirop@dellpc:/tmp$
```

Τα εισαγωγικά

- Τα απλά εισαγωγικά (‘single quotes’) άρουν την μεταχαρακτηρική ιδιότητα όλων των συμβόλων εκτός από τον εαυτό τους.
- Τα διπλά εισαγωγικά (“double quotes”) άρουν την μεταχαρακτηρική ιδιότητα όλων των συμβόλων εκτός από τον εαυτό τους, την ανάποδη κάθετο (backslash \ ) και το δολάριο ($).
- Τα ανάποδα εισαγωγικά (`back quotes`) προκαλούν την εκτέλεση της εντολής που περικλείουν.

Τα εισαγωγικά


```
bash-2.05a$ a='test    1'
bash-2.05a$ echo $a
test 1
bash-2.05a$ echo "$a"
test    1
bash-2.05a$ echo '$a'
$a
bash-2.05a$ echo "\$a"
$a
bash-2.05a$
```

Χρήση απλών & διπλών εισαγωγικών

- μέσα στα διπλά εισαγωγικά ερμηνεύονται οι μεταβλητές (μεταβλητές του shell αλλά και οι μεταβλητές περιβάλλοντος)
- μέσα στα μονά εισαγωγικά ΔΕΝ ερμηνεύονται οι μεταβλητές - δηλαδή αναιρείται η ειδική σημασία του χαρακτήρα $.

```
asidirop@aetos:/tmp$ echo "My host name is $HOSTNAME"
My host name is aetos
asidirop@aetos:/tmp$ echo 'My host name is $HOSTNAME'
My host name is $HOSTNAME
asidirop@aetos:/tmp$
```

Τα εισαγωγικά

- Το shell όταν βρίσκει το χαρακτήρα $, θεωρεί ότι αυτό που ακολουθεί (μέχρι κάποιον ειδικό χαρακτήρα ή κενό) είναι όνομα μεταβλητής (ή μεταβλητής περιβάλλοντος), και το αντικαθιστά με την τιμή του.

Συχνά χρησιμοποιούνται τα {} για να περικλείουν ονόματα μεταβλητών ώστε να ξεχωρίζουν από το string που ακολουθεί

```
bash-2.05a$ a='test    1'
bash-2.05a$ echo $abb

bash-2.05a$ echo "$a"bb
test    1bb
bash-2.05a$ echo ${a}bb
test 1bb
bash-2.05a$ echo $a"bb"
test 1bb
bash-2.05a$ echo "${a}bb"
test    1bb
```

Τα εισαγωγικά

- Η πράξη "συνένωση string" γίνεται αυτόματα, εάν 2 strings είναι κολλητά.

```
asidirop@aetos:/tmp$ a=This' is a 'test
asidirop@aetos:/tmp$ echo "$a"
This is a test
asidirop@aetos:/tmp$ b="This is test2""$a"'$a'
asidirop@aetos:/tmp$ echo $b
This is test2This is a test$a
asidirop@aetos:/tmp$
```

Τα εισαγωγικά

- Τα ανάποδα εισαγωγικά εκτελούν το string που περιέχουν (σαν εντολή), δεν εμφανίζεται τίποτα στην κανονική έξοδο, και «επιστρέφουν» ό,τι έχει στείλει η εντολή στην έξοδό της.

```
bash-2.05a$ date
Mon Nov 27 18:39:28 EET 2006
bash-2.05a$ a=`date`
bash-2.05a$ echo "$a"
Mon Nov 27 18:39:35 EET 2006
bash-2.05a$ a=`ls|wc`
bash-2.05a$ echo "$a"
     22              22             214
```

Τα εισαγωγικά


```
bash-2.05a$ b=`ls -l`
bash-2.05a$ echo $b
total 32 -rw-r--r-- 1 asidirop it 314 Jan 11 2003 cc -rwxr--
  r-- 1 asidirop it 77 Nov 20 19:17 file1 -rwxr-xr-x 1
  asidirop it 61 Nov 20 19:34 file2 -rw-r--r-- 1 asidirop it
  183 Jan 8 2003 list
bash-2.05a$ echo "$b"
total 32
-rw-r--r--    1 asidirop it          314 Jan 11  2003 cc
-rwxr--r--    1 asidirop it           77 Nov 20 19:17 file1
-rwxr-xr-x    1 asidirop it           61 Nov 20 19:34 file2
-rw-r--r--    1 asidirop it          183 Jan  8  2003 list
bash-2.05a$
```

- η έξοδος της εντολής ls -l, αποθηκεύεται σωστά στην μεταβλητή b, αρκεί να την χρησιμοποιήσουμε μέσα σε εισαγωγικά ώστε να μην ερμηνευτούν οι ειδικοί χαρακτήρες που αυτή τυχόν περιέχει.

Τα εισαγωγικά


```
bash-2.05a$ b=$(ls -l)
bash-2.05a$ echo "$b"
total 32
-rw-r--r--    1 asidirop it          314 Jan 11  2003 cc
-rwxr--r--    1 asidirop it           77 Nov 20 19:17 file1
-rwxr-xr-x    1 asidirop it           61 Nov 20 19:34 file2
-rw-r--r--    1 asidirop it          183 Jan  8  2003 list
bash-2.05a$
```

- Στο bash εναλλακτικά των ` (ανάποδων εισαγωγικών) μπορούμε να χρησιμοποιήσουμε και το **$(εντολή)**
- Το $() δεν δουλεύει όμως στην κανονική έκδοση του sh (Bourne Shell) - πλέον όμως αυτή η έκδοση δεν χρησιμοποιείται.

Αριθμητικές πράξεις

- Γενικά τα shells δεν ξέρουν να χειρίζονται αριθμούς.
- Οι πράξεις γίνονται με την βοήθεια εντολών.
- Η εντολή που χρησιμοποιείται για πράξεις ακεραίων είναι η expr.

```
asidirop@aetos:~$ expr 1 + 2
3
asidirop@aetos:~$ a=5
asidirop@aetos:~$ expr "$a" - 10
-5
asidirop@aetos:~$ b=4000
asidirop@aetos:~$ expr "$b" + "$a"
4005
asidirop@aetos:~$
```

Αριθμητικές πράξεις

- Προσοχή στην χρήση του πολλαπλασιασμού
- Το * είναι ειδικός χαρακτήρας για το shell, συνεπώς πρέπει να άρουμε την ειδική του σημασία.

```
asidirop@aetos:~$ expr 6 * 10
expr: syntax error
asidirop@aetos:~$ expr 6 \* 10
60
asidirop@aetos:~$ expr 6 '*' 10
60
asidirop@aetos:~$
```

Αριθμητικές πράξεις

- η expr τυπώνει το αποτέλεσμα των πράξεων στην τυπική έξοδο (stdout).
- Με την χρήση ` μπορούμε να εκχωρήσουμε το αποτέλεσμα των πράξεων σε μεταβλητή.

```
asidirop@aetos:~$ c=`expr 7 / 2`
asidirop@aetos:~$ echo "$c"
3
asidirop@aetos:~$ d=`expr "$a" + "$b"`
asidirop@aetos:~$ echo "$d"
4005
asidirop@aetos:~$
```

Αριθμητικές πράξεις

- Προσοχή: η expr περιμένει κάθε στοιχείο της αριθμητικής παράστασης ως διαφορετικό όρισμα.

```
asidirop@aetos:~$ expr 6+4
6+4
asidirop@aetos:~$ expr 6 +4
expr: syntax error
asidirop@aetos:~$
```

Αριθμητικές πράξεις

- Προσοχή: στην περίπτωση που κάποια μεταβλητή που θέλουμε να χρησιμοποιήσουμε με την expr δεν έχει οριστεί.
- αν δώσουμε ως όρισμα κάτι που δεν έχει οριστεί ή δεν είναι αριθμός, θα λάβουμε μήνυμα σφάλματος.

```
bash-2.05a$ a=1
bash-2.05a$ expr $a + $w
Syntax error
bash-2.05a$ expr 1 +
Syntax error
bash-2.05a$ expr "$a" + "$w"
non-numeric argument
bash-2.05a$ expr 0"$a" + 0"$w"
1
bash-2.05a$ k=`expr $c + $w`
Syntax error
bash-2.05a$ echo $k

bash-2.05a$
```

Αριθμητικές πράξεις

- **Το bash γνωρίζει να κάνει αριθμητικές πράξεις**
- δεν έχουμε πρόβλημα με τα κενά
- δεν έχουμε πρόβλημα αν μια μεταβλητή δεν είναι ορισμένη (αρκεί να μην χρησιμοποιούμε το $)
- δεν έχουμε πρόβλημα αν μια μεταβλητή δεν περιέχει αριθμό (θεωρείται 0)

```
asidirop@aetos:~$ a=$((5+4))
asidirop@aetos:~$ echo $a
9
asidirop@aetos:~$ a=$((a+4))
asidirop@aetos:~$ echo $a
13
asidirop@aetos:~$ a=$((a+w))
asidirop@aetos:~$ echo $a
13
asidirop@aetos:~$ a=$((a+$w))
-bash: a+: syntax error: operand
  expected (error token is "+")
asidirop@aetos:~$
asidirop@aetos:~$ w='test'
asidirop@aetos:~$ a=$((a+w))
asidirop@aetos:~$ echo $a
13
```

Αριθμητικές πράξεις

- **Προσοχή όμως στην εξής περίπτωση:**
- Το $ ερμηνεύεται πριν από την εκτέλεση της πράξης. Άρα το $b θα αντικατασταθεί με το test και το $a με το 13. Άρα είναι σαν να γράφουμε:
  `c=$((13+test))`
  και το test είναι μεταβλητή που περιέχει τιμή!!!!

```
asidirop@aetos:~$ a=13
asidirop@aetos:~$ b='test'
asidirop@aetos:~$ test='1000'
asidirop@aetos:~$ c=$(($a+$b))
asidirop@aetos:~$ echo $c
1013
asidirop@aetos:~$
```

Αριθμητικές πράξεις

- **συμπέρασμα: χρησιμοποιείτε την δυνατότητα του bash** αρκεί το script σας να ξεκινάει με
  `#!/bin/bash`
  και όχι με
  #!/bin/sh (αυτό μπορεί να μας δημιουργήσει προβλήματα αν μεταφέρουμε το script μας σε άλλο σύστημα)