Εισαγωγή στα Λειτουργικά Συστήματα

SET ΔΙΑΦΑΝΕΙΩΝ 12

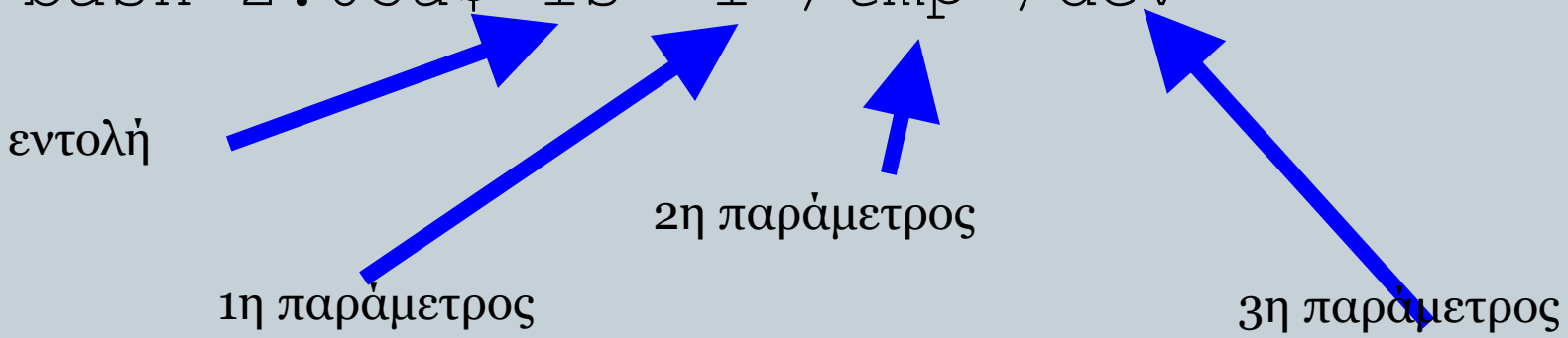
SHELL SCRIPTS ARGUMENTS + FOR

ΑΝΤΩΝΗΣ ΣΙΔΗΡΟΠΟΥΛΟΣ

Χρήση Command-Line Parameters

- Κάθε φορά που το κέλυφος διερμηνεύει μια εντολή συνδέει ονόματα μεταβλητών σε κάθε παράμετρο της γραμμής εντολών. Ως παράμετροι της γραμμής εντολών θεωρούνται ακολουθίες χαρακτήρων διαχωριζόμενες με κενά ή tab. (χρησιμοποιείστε εισαγωγικά αν πρόκειται χαρακτήρες με ενδιάμεσα κενά να αντιπροσωπεύουν ένα στοιχείο).

```
bash-2.05a$ ls -l /tmp /dev
```



Shell programming – διαχείριση μεταβλητών

- Ειδικές μεταβλητές φλοιού

όνομα	περιγραφή
`$1 - $9`	Οι παράμετροι στο script
`$0`	το όνομα της εντολής (όπως το έδωσε ο χρήστης)
`$#`	το πλήθος των παραμέτρων
`$*`	ένα string που περιλαμβάνει όλα τις παραμέτρους
`$@`	το ίδιο με το $*, εκτός αν χρησιμοποιούνται εισαγωγικά (είναι πίνακας)
`$$`	ο αριθμός διεργασίας (PID) του φλοιού
`$!`	Ο αριθμός διεργασίας της διεργασίας που εκτελείται (εκτελέστηκε) στο παρασκήνιο
`$?`	Η κατάσταση εξόδου (exit status) της εντολής που εκτελέστηκε τελευταία .
`${10}-..`	Στο bash υπάρχουν και τα ορίσματα >=10 χρησιμοποιώντας {}.

$* επεκτείνεται στις παραμέτρους θέσης (positional parameters), ξεκινώντας από την πρώτη. Όταν η επέκταση γίνεται χρησιμοποιώντας διπλά εισαγωγικά, υλοποιείται με μια μόνο λέξη που περιλαμβάνει την τιμή κάθε παραμέτρου που διαχωρίζεται από τις υπόλοιπες τιμές με διαχωριστικό που αντιστοιχεί στον πρώτο χαρακτήρα της ειδικής μεταβλητής 'IFS'.

$@ επεκτείνεται στις παραμέτρους θέσης , ξεκινώντας από την πρώτη. Όταν η επέκταση γίνεται χρησιμοποιώντας διπλά εισαγωγικά, κάθε παράμετρος επεκτείνεται σε μια ξεχωριστή λέξη.

$# αντιστοιχεί στον αριθμό των παραμέτρων θέσης στο δεκαδικό σύστημα αρίθμησης.

$0 αντιστοιχεί στο όνομα του φλοιού ή του σεναρίου φλοιού.

Παραδείγματα


```
bash-2.05a$ ./test_params 22 33333
param 0=./test_params
param 1=22
param 2=33333
bash-2.05a$ ./test_params "2 3 4"
param 0=./test_params
param 1=2 3 4
param 2=
bash-2.05a$ ./test_params "2 3 4" "5 6 7"
param 0=./test_params
param 1=2 3 4
param 2=5 6 7
bash-2.05a$ ./test_params 2 3 4 5 6 7
param 0=./test_params
param 1=2
param 2=3
```

αρχείο test_params

```
#!/bin/bash

echo "param 0=$0"
echo "param 1=$1"
echo "param 2=$2"
```

Παραδείγματα


```
bash-2.05a$ ./test_params 2 3 4 5 6 7
param 0=./test_params
param 1=2
param 2=3
bash-2.05a$ ~/test/test_params 2 3 4 5
param
  0=/usr/people/staff/ektaktoi/it/asidirop/test/
  test_params
param 1=2
param 2=3
bash-2.05a$
```

αρχείο test_params

```
#!/bin/bash

echo "param 0=$0"
echo "param 1=$1"
echo "param 2=$2"
```

- Το $0 περιέχει το όνομα εντολής όπως το έδωσε ο χρήστης (αφού βέβαια ερμηνευτούν οι ειδικοί χαρακτήρες)

Παραδείγματα

FILE: testparms

```
#!/bin/bash
# FILE: testparms
echo "all are $1 $2 $3 $4 $5 $6 $7 $8 $9 $10 $11 $12"
echo "asteraki is $*"
echo "diesi is $#"
```

```
$./testparms a b c d e f g h i j k l
all are a b c d e f g h i a0 a1 a2
asteraki is a b c d e f g h i j k l
diesi is 12
```

- Το $10 σημαίνει $1 και κολλητά ο χαρακτήρας "0"
- μπορούμε να χρησιμοποιήσουμε μέχρι το $9 (ένατη παράμετρος)
- μπορούμε να χρησιμοποιήσουμε τις επόμενες χρησιμοποιώντας loop.

File: fs

```
#!/bin/bash

echo "$1 status is:"
stat "$1"
touch "$1"
echo "after the touch, $1 became:"
stat "$1"
```

```
bash-2.05a$ ./fs cc
cc status is:
cc:
        inode 21316263; dev 308; links 1; size 314
        regular; mode is rw-r--r--; uid 1143 (asidirop); gid 1000 (it)
        projid 0        st_fstype: xfs
        change time - Sat Jan 11 20:07:07 2003 <1042308427>
        access time - Fri Oct 20 03:36:28 2006 <1161304588>
        modify time - Sat Jan 11 20:07:07 2003 <1042308427>
after the touch, cc became:
cc:
        inode 21316263; dev 308; links 1; size 314
        regular; mode is rw-r--r--; uid 1143 (asidirop); gid 1000 (it)
        projid 0        st_fstype: xfs
        change time - Mon Nov 27 19:36:17 2006 <1164648977>
        access time - Mon Nov 27 19:36:17 2006 <1164648977>
        modify time - Mon Nov 27 19:36:17 2006 <1164648977>
bash-2.05a$
```

Εντολές προγραμματισμού στο κέλυφος


```
for i in λίστα λέξεων;
do
     εντολές, το $i παίρνει διαδοχικά τις
  τιμές της λίστας λέξεων
done
```

Παράδειγμα :

```
 #!/bin/bash
for i in word1 word2 word3; do
    echo $i
done
```

Παράδειγμα

File: test2

```
#!/bin/sh

echo "first test"
for i in 1 2 3 4; do
        echo "i is $i"
done
echo "second test"
for i in "1 2 3 4"; do
        echo "i is $i"
done
echo "third test"
for i in "1 2" "3" 4; do
        echo "i is $i"
done
```

```
bash-2.05a$ ./test2
first test
i is 1
i is 2
i is 3
i is 4
second test
i is 1 2 3 4
third test
i is 1 2
i is 3
i is 4
bash-2.05a$
```

Παράδειγμα: χειρισμός παραμέτρων


```
bash-2.05a$ ./sc1 aaaa 22222 ccc
params: 3
i is aaaa
i is 22222
i is ccc
param1 is aaaa
param2 is 22222
param3 is ccc
bash-2.05a$ ./sc1 aaaa "22222 ccc"
params: 2
i is aaaa
i is 22222
i is ccc
param1 is aaaa
param2 is 22222 ccc
param3 is
bash-2.05a$
```

File:sc1

```
#!/bin/bash
echo "params: $#"
for i in $@;do
    echo "i is $i"
done
echo "param1 is $1"
echo "param2 is $2"
echo "param3 is $3"
```

- όταν δώσουμε μέσα σε ένα όρισμα κάποιον ειδικό χαρακτήρα (πχ: space), τότε η χρήση της for γίνεται προβληματική.

Παράδειγμα: χειρισμός παραμέτρων


```
bash-2.05a$ ./sc1 aaaa "*"
params: 2
i is aaaa
i is 20112xeim-adopse.tar.bz2
i is admin_menu-7.x-3.0-rc2.tar.gz
i is arx
i is arxeio2
i is asd
i is ask3erg4
param1 is aaaa
param2 is *
param3 is
bash-2.05a$
```

File:sc1

```
#!/bin/bash
echo "params: $#"
for i in $@;do
    echo "i is $i"
done
echo "param1 is $1"
echo "param2 is $2"
echo "param3 is $3"
```

- όταν δώσουμε μέσα σε ένα όρισμα κάποιον ειδικό χαρακτήρα (πχ: *), τότε η χρήση της for γίνεται προβληματική.

Παράδειγμα


```
bash-2.05a$ ./sc2 aaaa 22222 ccc
params: 3
i is aaaa
i is 22222
i is ccc
param1 is aaaa
param2 is 22222
param3 is ccc
bash-2.05a$ ./sc2 aaaa "22222 ccc"
params: 2
i is aaaa
i is 22222
i is ccc
param1 is aaaa
param2 is 22222 ccc
param3 is
bash-2.05a$
```

File:sc2

```
#!/bin/bash
echo "params: $#"
for i in $*;do
    echo "i is $i"
done
echo "param1 is $1"
echo "param2 is $2"
echo "param3 is $3"
```

- Το ίδιο πρόβλημα με τους ειδικούς χαρακτήρες έχουμε και με την χρήση του $*.

Παράδειγμα


```
bash-2.05a$ ./sc3 aaaa 22222 ccc
params: 3
i is aaaa 22222 ccc
param1 is aaaa
param2 is 22222
param3 is ccc
bash-2.05a$ ./sc3 aaaa "22222 ccc"
params: 2
i is aaaa 22222 ccc
param1 is aaaa
param2 is 22222 ccc
param3 is
bash-2.05a$
```

File:sc3

```
#!/bin/bash
echo "params: $#"
for i in "$*";do
    echo "i is $i"
done
echo "param1 is $1"
echo "param2 is $2"
echo "param3 is $3"
```

- με την χρήση του $* μέσα σε εισαγωγικά "$*" γίνεται χειρότερη η κατάσταση διότι ο βρόγχος θα κάνει ΜΙΑ επανάληψη.

Παράδειγμα


```
bash-2.05a$ ./sc4 aaaa 22222 ccc
params: 3
i is aaaa
i is 22222
i is ccc
param1 is aaaa
param2 is 22222
param3 is ccc
bash-2.05a$ ./sc4 aaaa "22222 ccc"
params: 2
i is aaaa
i is 22222 ccc
param1 is aaaa
param2 is 22222 ccc
param3 is
bash-2.05a$
```

File:sc4

```
#!/bin/bash
echo "params: $#"
for i in "$@";do
    echo "i is $i"
done
echo "param1 is $1"
echo "param2 is $2"
echo "param3 is $3"
```

- η σωστή μεθοδος για να διατρέξουμε όλες τις παραμέτρους είναι με χρήση του $@ μέσα σε εισαγωγικά "$@".
- τότε κάθε στοιχείο του πίνακα μπαίνει σε "

Παράδειγμα

- Το script ff, μπορεί να δεχθεί απεριόριστο πλήθος παραμέτρων. κάθε παράμετρος υποτίθεται ότι εκφράζει όνομα αρχείου.
- Για κάθε ένα εκτελούμε την ls -l, wc -l και την file.

File:ff

```
#!/bin/bash
for i in "$@"; do
   echo ------------------
   ls -l "$i"
   wc -l "$i"
   file "$i"
   echo ------------------
Done
```

```
bash-2.05a$ ./ff file1 file2
-----------------------
-rwxr--r--    1 asidirop it           77 Nov 20 19:17 file1
              4 file1
file1:          /bin/sh -x script text
-----------------------
-----------------------
-rwxr-xr-x    1 asidirop it           61 Nov 20 19:34 file2
              7 file2
file2:          /bin/sh script text
-----------------------
```

1η επανάληψη i="file1"

2η επανάληψη i="file2"

Παράδειγμα

- εάν στο script μας δώσουμε όρισμα που περιέχει wildcard (πχ file*), τότε το wildcard (ή όποιος άλλος ειδικός χαρακτήρας) ερμηνεύεται από το shell και όχι από το script μας.
- εσωτερικά από το script δεν υπάρχει τρόπος να βρούμε ότι ο χρήστης έδωσε
./ff file* και όχι
./ff file1 file2

File:ff

```
#!/bin/bash
for i in "$@"; do
   echo ------------------
   ls -l $i
   wc -l $i
   file $i
   echo ------------------
done
```

1η επανάληψη i="file1"

2η επανάληψη i="file2"

```
bash-2.05a$ ./ff file*
------------------------
-rwxr--r--    1 asidirop it             77 Nov 20 19:17 file1
              4 file1
file1:          /bin/sh -x script text
------------------------
------------------------
-rwxr-xr-x    1 asidirop it             61 Nov 20 19:34 file2
              7 file2
file2:          /bin/sh script text
------------------------
```

Παράδειγμα

- εάν στο script μας δώσουμε όρισμα που περιέχει ειδικό χαρακτήρα, αυτός ερμηνεύεται από το shell πριν εκτελεστεί το script.

1η επανάληψη
i="
/home/staff/ektaktoi/asidirop/file1"

2η επανάληψη
i="
/home/staff/ektaktoi/asidirop/file_x"

2η επανάληψη
i="./file3"

```
asidirop@aetos:/tmp$ ./ff ~/fi* ./file3
-----------------
-rw-r--r-- 2 asidirop conit 0 Apr 27 02:37
/home/staff/ektaktoi/asidirop/file1
0 /home/staff/ektaktoi/asidirop/file1
/home/staff/ektaktoi/asidirop/file1: empty
-----------------
-----------------
-rw-r--r-- 1 asidirop conit 0 May 14 14:02
/home/staff/ektaktoi/asidirop/file_x
0 /home/staff/ektaktoi/asidirop/file_x
/home/staff/ektaktoi/asidirop/file_x: empty
-----------------
-----------------
-rw-r--r-- 1 asidirop conit 422 Apr  9 09:58 ./file3
16 ./file3
./file3: ASCII text
-----------------
asidirop@aetos:/tmp$
```

for

- τα περισσότερα shells δεν μπορούν να χειριστούν τους αριθμούς.
- Η for διατρέχει την μεταβλητή στην λίστα παραμέτρων (λέξεων) που τις δίνεται.
- Το bash μπορεί να χειριστεί αριθμούς και έχει ειδική σύνταξη της for όπως στις γνωστές γλώσσες προγραμματισμού C, Java, κτλ.
- σύνταξη:

```
for ((expression; expression; expression))
do
    commands
done
```

- Προσοχή στις διπλές παρενθέσεις.
- Το παρακάτω ισχύει μόνο για το bash
- μέσα στις (( )) ισχύει συντακτικό παρόμοιο με την C,java.
- Οι περισσότεροι ειδικοί χαρακτήρες τους shell, χάνουν την σημασία τους μέσα στις (( )).

```
asidirop@aetos:/tmp$ for((i=1;i<10;i++))  do echo $i; done
1
2
3
4
5
6
7
8
9
```

Πράξεις μέσα σε (( )) για το bash

`id++ id--`	variable post-increment and post-decrement
`++id --id`	variable pre-increment and pre-decrement
`- +`	unary minus and plus
`! ~`	logical and bitwise negation
`**`	exponentiation
`* ,/ , %`	multiplication, division, remainder
`+, -`	addition, subtraction
`<< , >>`	left and right bitwise shifts
`<= ,>=, <, >`	comparison
`==, !=`	equality and inequality
`&, ^, |`	bitwise AND, bitwise exclusive OR, bitwise OR
`&&, ||`	logical AND, logical OR
`expr?expr:expr`	conditional operator
`= *= /= %= += -= <<= >>= &= ^= |=`	assignment