

Εισαγωγή στα Λειτουργικά Συστήματα



SET ΔΙΑΦΑΝΕΙΩΝ 13

SHELL SCRIPTS: WHILE, UNTIL, CASE

ΑΝΤΩΝΗΣ ΣΙΔΗΡΟΠΟΥΛΟΣ

η εντολή while

2

- Σύνταξη (σύμφωνα με το manual του bash):
`while list; do list; done`
- δηλαδή:
`while εντολή-ελέγχου ; do`
 `εντολές`
`done`
- η εντολή ελέγχου ερμηνεύεται με τον ίδιο τρόπο που ερμηνεύεται η εντολή ελέγχου σε μια if.

η εντολή while

3

- συνήθως, ως έλεγχος στην while χρησιμοποιείται έλεγχος παρόμοιος με την if.

file: while_test

```
#!/bin/bash
number=0
while [ "$number" -lt 10 ]; do
    echo "Number = $number"
    number=$((number + 1))
done
```

```
asidirop@dellpc:/tmp$ ./while_test
Number = 0
Number = 1
Number = 2
Number = 3
Number = 4
Number = 5
Number = 6
Number = 7
Number = 8
Number = 9
asidirop@dellpc:/tmp$
```

η εντολή while

4

- Η [, μπορεί να αντικατασταθεί με τις (()).

```
asidirop@dellpc:/tmp$ ./while_test2
Number = 0
Number = 1
Number = 2
Number = 3
Number = 4
Number = 5
Number = 6
Number = 7
Number = 8
Number = 9
asidirop@dellpc:/tmp$
```

file: while_test2

```
#!/bin/bash
number=0
while ((number<10)); do
    echo "Number = $number"
    number=$((number + 1))
done
```

- Σε αυτήν την περίπτωση θα μπορούσε να είχε παραλειφθεί η αρχικοποίηση της μεταβλητής (καλό είναι όμως να γίνεται πάντα αρχικοποίηση)
- το πρώτο echo όμως θα εμφάνιζε "" και όχι "0".

η εντολή while

5

- Για την αύξηση κατά ένα, θα μπορούσε να είχε χρησιμοποιηθεί η αντίστοιχη συντομογραφία του bash σε "arithmetic mode"

```
asidirop@dellpc:/tmp$ ./while_test3
Number = 0
Number = 1
Number = 2
Number = 3
Number = 4
Number = 5
Number = 6
Number = 7
Number = 8
Number = 9
asidirop@dellpc:/tmp$
```

file: while_test3

```
#!/bin/bash
number=0
while ((number<10)); do
    echo "Number = $number"
    ((number++))
done
```

η εντολή while: παράδειγμα

6

- στην περίπτωση "λανθασμένης" απάντησης του χρήστη, θέλουμε να επαναλάβουμε το loop.

file: while_example

```
#!/bin/bash

ok=0
while ((ok==0)); do
    echo -n "Do you want to continue? (y/n) "
    read a
    if [ "$a" = "y" -o "$a" = "Y" ] ; then
        ok=1
    elif [ "$a" = "n" -o "$a" = "N" ] ; then
        echo "exiting"
        exit
    else
        echo "Wrong answer... please type y or n"
    fi
done
echo "continue"
```

η εντολή while: παράδειγμα

7

```
asidirop@dellpc:/tmp$ ./while_example
Do you want to continue? (y/n) q
Wrong answer... please type y or n
Do you want to continue? (y/n) n
exiting
asidirop@dellpc:/tmp$ ./while_example
Do you want to continue? (y/n)
Wrong answer... please type y or n
Do you want to continue? (y/n) y
continue
asidirop@dellpc:/tmp$
```

πατήσαμε
q+Enter

πατήσαμε
n+Enter

πατήσαμε
Enter
(άρα άδειο
string)

πατήσαμε
y+Enter

η εντολή until

8

- Σύνταξη (σύμφωνα με το manual του bash):
`until list; do list; done`
- δηλαδή:
`until εντολή-ελέγχου ; do`
 `εντολές`
`done`
- η εντολή ελέγχου ερμηνεύεται με τον ίδιο τρόπο που ερμηνεύεται η εντολή ελέγχου σε μια if ή while.
- Ο έλεγχος σε σχέση με την while είναι ο αντίστροφος: θα επαναλαμβάνεται το loop όσο επιστρέφεται false.

η εντολή until

9

- όταν θα γίνει `number==10` τότε θα βγούμε από τον βρόγχο.

file: until_test3

```
#!/bin/bash
number=0
until ((number==10)); do
    echo "Number = $number"
    ((number++))
done
```

```
asidirop@dellpc:/tmp$ ./until_test3
Number = 0
Number = 1
Number = 2
Number = 3
Number = 4
Number = 5
Number = 6
Number = 7
Number = 8
Number = 9
asidirop@dellpc:/tmp$
```

η εντολή until: παράδειγμα

10

- το παράδειγμα "while_example" με χρήση της until.

file: until_example

```
#!/bin/bash

ok=0
until ((ok!=0)); do
    echo -n "Do you want to continue? (y/n) "
    read a
    if [ "$a" = "y" -o "$a" = "Y" ] ; then
        ok=1
    elif [ "$a" = "n" -o "$a" = "N" ] ; then
        echo "exiting"
        exit
    else
        echo "Wrong answer... please type y or n"
    fi
done
echo "continue"
```

η εντολή case

11

- Σύνταξη (σύμφωνα με το manual του bash):

```
case word in [ ([] pattern [ | pattern ] ... ) list ;;  
] ... esac
```

δηλαδή:

```
case value in  
pattern1) command  
          anothercommand  
          ;;  
pattern2) command  
          anothercommand  
          ;;  
esac
```

- Το block της case, κλείνει με "esac" (ανάποδα όπως και στην if ... fi)
- Οι περιπτώσεις χωρίζονται με ;;
- Η χρήση της case συμφέρει έναντι πολλαπλών if.
- Το pattern είναι string (δεν γίνεται αριθμητική σύγκριση)

η εντολή case: παράδειγμα

12

- Καλύτερη αναγνωσιμότητα σε σχέση με πολλαπλή if.

```
file: case_example1
```

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

```
ok=0
```

```
while ((ok==0)); do
```

```
    echo -n "Do you want to continue? (y/n) "
```

```
    read a
```

```
    case "$a" in
```

```
        y) ok=1 ;;
```

```
        n) echo "exiting"
```

```
            exit ;;
```

```
        *) echo "Wrong answer... please type y or n" ;;
```

```
    esac
```

```
done
```

```
echo "continue"
```

η εντολή case: παράδειγμα

13

- Στο pattern χρησιμοποιούνται οι ίδιοι κανόνες με την αντικατάσταση των wild-cards (using the same matching rules as for pathname expansion - αναφέρεται στο manual)

ένας
χαρακτήρας
από τους [Yy]

οποιοσδήποτε
συνδυασμός
χαρακτήρων.
Θα φτάσουμε εδώ
μόνο αν καμία από
τις προηγούμενες
περιπτώσεις δεν
ταιριάζει.

file: case_example2

```
#!/bin/bash

ok=0
while ((ok==0)); do
    echo -n "Do you want to continue? (y/n) "
    read a
    case "$a" in
        [yY]) ok=1 ;;
        [nN]) echo "exiting"
                exit ;;
        *) echo "Wrong answer... please type y or n" ;;
    esac
done
echo "continue"
```

η εντολή case: παράδειγμα

14

- υποστηρίζεται και η πράξη OR (|) στο pattern.

ή Y ή y
ή yes ή Yes

ή n ή N
ή no ή No

file: case_example3

```
#!/bin/bash

ok=0
while ((ok==0)); do
    echo -n "Do you want to continue? (y/n) "
    read a
    case "$a" in
        [yY]|[Yy]es) ok=1 ;;
        [nN]|[Nn]o) echo "exiting"
                     exit ;;
        *) echo "Wrong answer... please type y or n" ;;
    esac
done
echo "continue"
```

η εντολή case: παράδειγμα

15

- Το * σημαίνει οποιοσδήποτε συνδυασμός χαρακτήρων.

οτιδήποτε ξεκινά
με y ή Y

οτιδήποτε ξεκινά
με n ή N

file: case_example4

```
#!/bin/bash

ok=0
while ((ok==0)); do
    echo -n "Do you want to continue? (y/n) "
    read a
    case "$a" in
        [yY]*) ok=1 ;;
        [nN]*) echo "exiting"
                exit ;;
        *) echo "Wrong answer... please type y or n" ;;
    esac
done
echo "continue"
```

Η εντολή shift

16

- Η εντολή shift μετακινεί τα στοιχεία του πίνακα ορισμάτων. προαιρετικά δέχεται όρισμα το πλήθος των στοιχείων μετακίνησης.
- Εάν δεν δοθεί όρισμα, τότε γίνεται μετακίνηση κατά ένα στοιχείο.
- Το πρώτο στοιχείο από τον πίνακα ορισμάτων χάνεται (και δεν υπάρχει τρόπος να ανακτηθεί).
- Άρα πριν χρησιμοποιηθεί η shift θα πρέπει να έχουμε χειριστεί τα στοιχεία που θα χαθούν.

file: shift_test

```
#!/bin/bash

echo "arg array: "$@"
echo "arg array: $*"
echo "arg array length: $#"
```



```
shift 2

echo "After the shift 2"
echo "-----"
echo "arg array: "$@"
echo "arg array: $*"
echo "arg array length: $#"
```


Η εντολή shift

17

```
asidirop@dellpc:/tmp$ ./shift_test aaa bbb ccc ddd
arg array: aaa bbb ccc ddd
arg array: aaa bbb ccc ddd
arg array length: 4
After the shift 2
-----
arg array: ccc ddd
arg array: ccc ddd
arg array length: 2
asidirop@dellpc:/tmp$
```

file: shift_test

```
#!/bin/bash

echo "arg array: "$@"
echo "arg array: $*"
echo "arg array length: $#"
```

shift 2

```
echo "After the shift 2"
echo "-----"
echo "arg array: "$@"
echo "arg array: $*"
echo "arg array length: $#"
```

Η εντολή shift

18

- Η εντολή shift χρησιμοποιείται σε συνδυασμό με την while για να διατρέξουμε την λίστα ορισμάτων που έδωσε ο χρήστης (εναλλακτικά της for)

```
asidirop@dellpc:/tmp$ ./shift_while aaa bbb ccc  
ddd 'eeee * eeee'  
Number of args: 5  
Got an arg: aaa  
Got an arg: bbb  
Got an arg: ccc  
Got an arg: ddd  
Got an arg: eeee * eeee  
asidirop@dellpc:/tmp$
```

file: shift_test

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

```
echo "Number of args: $#"
```

```
while (($#>0)) ; do
```

```
    echo "Got an arg: $1"
```

```
    shift
```

```
done
```

παράδειγμα

19

- Δημιουργείστε ένα script με όνομα `files_info`. Το script θα πρέπει να δέχεται τα ορίσματα:
 - `-t` (βρες τους τύπους αρχείων χρησιμοποιώντας την `file`)
 - `-d` (εμφάνισε την έξοδο από την εντολή `stat`)

Παράδειγμα – λύση 1

file: args_test

```
#!/bin/bash

use_command_file=0
use_command_stat=0
for i in "$@"; do
    case "$i" in
        -t) use_command_file=1;;
        -d) use_command_stat=1;;
        *) echo "Unknown param $1" 1>&2
           echo "Valid params are: [-t] [-d]" 1>&2
           exit 1
           ;;
    esac
done
echo "You have set the params:
use_command_file=$use_command_file
use_command_stat=$use_command_stat"
if ((use_command_file==1)); then
    file *
fi
if ((use_command_stat==1)); then
    stat *
fi
```

Παράδειγμα – λύση 2

file: args_test

```
#!/bin/bash

use_command_file=0
use_command_stat=0
while (( $#> 0 )) ; do
    case "$1" in
        -t) use_command_file=1;;
        -d) use_command_stat=1;;
        *) echo "Unknown param $1" 1>&2
           echo "Valid params are: [-t] [-d]" 1>&2
           exit 1
           ;;
    esac
    shift 1
done
echo "You have set the params:
use_command_file=$use_command_file
use_command_stat=$use_command_stat"
if ((use_command_file==1)); then
    file *
fi
if ((use_command_stat==1)); then
    stat *
fi
```

Παράδειγμα 2

22

- Η προηγούμενη μέθοδος για να διατρέξουμε τον πίνακα ορισμάτων, προτιμάται σε σχέση με την for στην περίπτωση που θέλουμε να χειριστούμε πολλά ορίσματα μαζί. πχ:
- Δημιουργείστε ένα script με όνομα `files_info`. Το script θα πρέπει να δέχεται τα ορίσματα:
 - `-t` (βρες τους τύπους αρχείων χρησιμοποιώντας την `file`)
 - `-d` (εμφάνισε την έξοδο από την εντολή `stat`)
 - `-fi file1` (μόνο το αρχείο `file1` - αν δοθεί το όρισμα `-fi` τότε ακολουθείται από όνομα αρχείου)

παράδειγμα

file: args_test

```
#!/bin/bash

use_command_file=0
use_command_stat=0
file=""
while (( $# > 0 )) ; do
    case "$1" in
        -t) use_command_file=1;;
        -d) use_command_stat=1;;
        -fi) shift 1; file="$1";;
        *) echo "Unknown param $1" 1>&2
           echo "Valid params are: [-t] [-d] [-fi file]" 1>&2
           exit 1
           ;;
    esac
    shift 1
done
echo "You have set the params:
use_command_file=$use_command_file
use_command_stat=$use_command_stat
file=$file"
```

μόλις βρούμε το
όρισμα -fi,
χειριζόμαστε και
το επόμενο
όρισμα. Κάτι
τέτοιο θα ήταν
πολύ δύσκολο να
υλοποιηθεί με την
for

παράδειγμα

24

```
asidirop@dellpc:/tmp$ ./args_test -t -d
You have set the params:
use_command_file=1
use_command_stat=1
file=
asidirop@dellpc:/tmp$ ./args_test -d -t
You have set the params:
use_command_file=1
use_command_stat=1
file=
asidirop@dellpc:/tmp$ ./args_test -fi test -d -t
You have set the params:
use_command_file=1
use_command_stat=1
file=test
asidirop@dellpc:/tmp$ ./args_test test -d -t
Unknown param test
Valid params are: [-t] [-d] [-fi file]
asidirop@dellpc:/tmp$
```

δεν παίζει ρόλο η
σειρά των
ορισμάτων

λάθος όρισμα. --
το χειριζόμαστε.

παράδειγμα

25

- Bug1:
 - `-fi) shift 1; file="$1";;`
 - Δεν ελέγχουμε αν υπάρχει επόμενο όρισμα
 - Θα έπρεπε πριν από την `shift` αν υπάρχει επόμενο στοιχείο στον πίνακα ορισμάτων.

παράδειγμα

26

```
-fi) shift 1; file="$1";;
```



```
-fi) if (($#>1)); then
```

```
    shift
```

```
    file="$1";;
```

```
else
```

```
    echo "Error: -fi must be followed by a filename" 1>&2
```

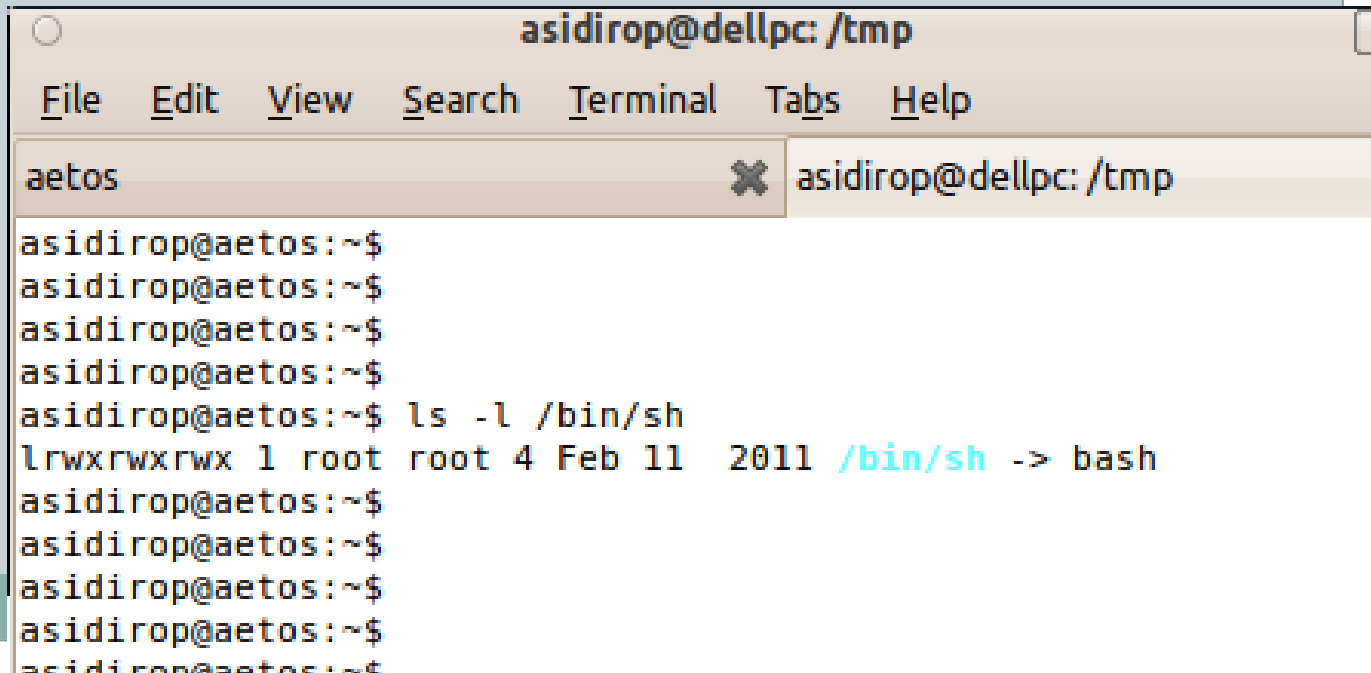
```
    exit 2
```

```
fi ;;
```

Συμβατότητα κελυφών - προσοχή

27

- όπως έχει αναφερθεί δεν είναι όλα τα shells συμβατά μεταξύ τους.
- Το bash είναι συμβατό με sh
- Το sh δεν είναι συμβατό με το bash.
- Σε αρκετά συστήματα, δεν χρησιμοποιείται το sh, αλλά το bash ως βελτιωμένο αντικατάστατο. πχ στον aetos (debian)
- πρακτικά (στον aetos) είτε χρησιμοποιήσουμε bash είτε sh είναι το ίδιο.

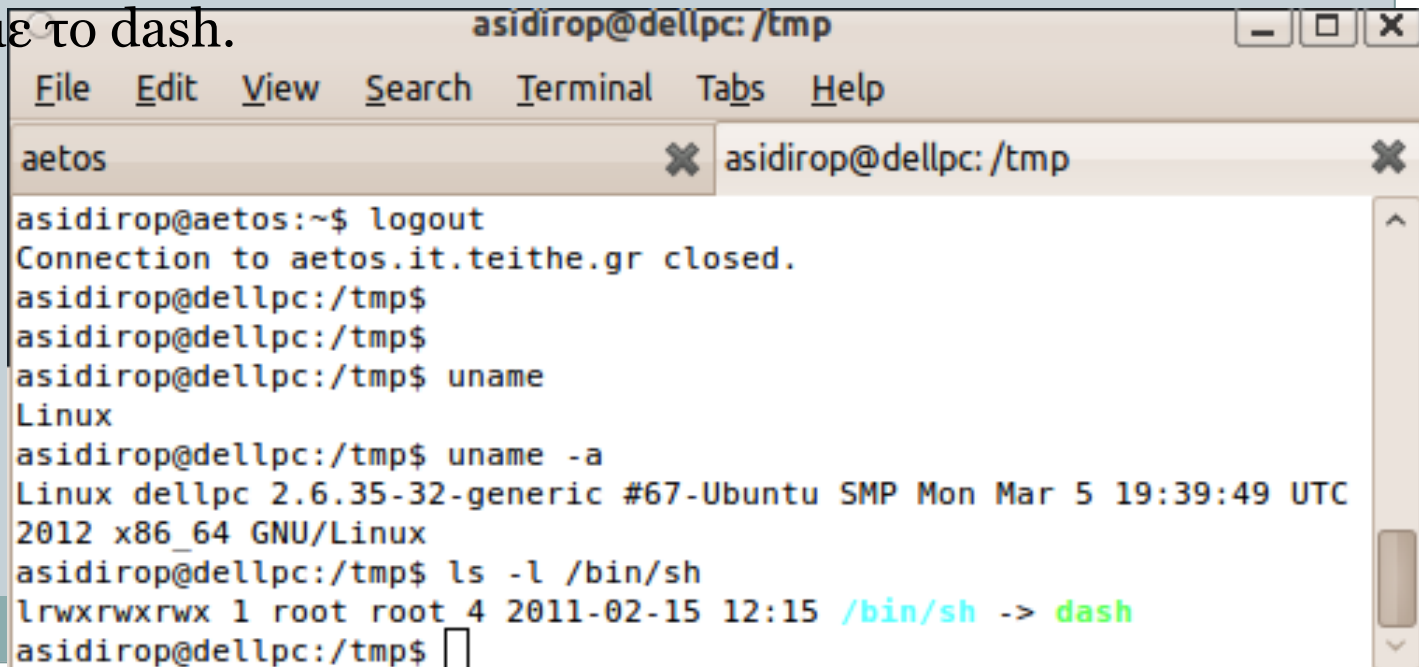


```
asidirop@dellpc: /tmp
File Edit View Search Terminal Tabs Help
aetos x asidirop@dellpc: /tmp
asidirop@aetos:~$
asidirop@aetos:~$
asidirop@aetos:~$
asidirop@aetos:~$
asidirop@aetos:~$ ls -l /bin/sh
lrwxrwxrwx 1 root root 4 Feb 11 2011 /bin/sh -> bash
asidirop@aetos:~$
asidirop@aetos:~$
asidirop@aetos:~$
asidirop@aetos:~$
asidirop@aetos:~$
```

Συμβατότητα κελυφών - προσοχή

28

- Όμως σε άλλα συστήματα μπορεί το sh να μην αντικαθίσταται από το bash αλλά από άλλο shell συμβατό με το sh (ή και κανένα - να υπάρχει δηλαδή η έκδοση sh)
- παράδειγμα σε ubuntu 10.10 το sh έχει αντικατασταθεί από το dash.
- άρα όταν χρησιμοποιήσουμε το sh στον aeto, πρακτικά χρησιμοποιούμε το bash, όταν χρησιμοποιήσουμε το sh σε ubuntu 10.10 πρακτικά χρησιμοποιούμε το dash.



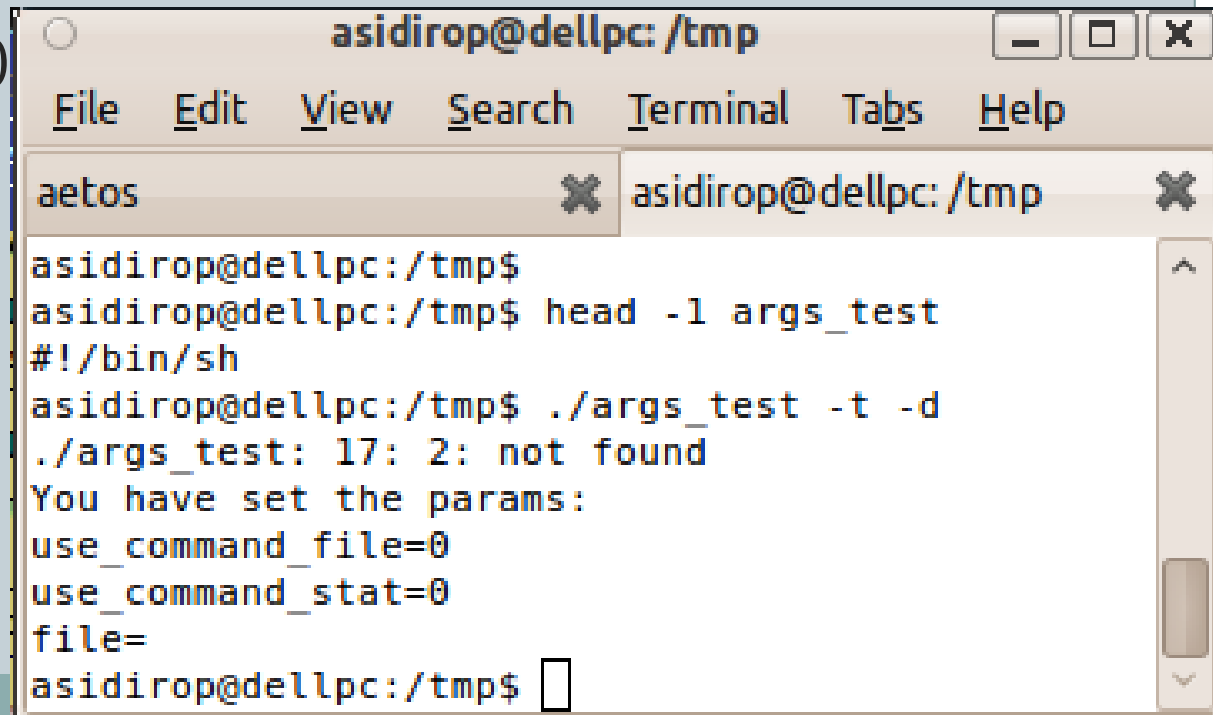
The screenshot shows a terminal window titled 'asidirop@dellpc: /tmp'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', 'Tabs', and 'Help'. Below the menu bar, there are two tabs: 'aetos' and 'asidirop@dellpc: /tmp'. The terminal content shows the following commands and output:

```
asidirop@aetos:~$ logout
Connection to aetos.it.teithe.gr closed.
asidirop@dellpc:/tmp$
asidirop@dellpc:/tmp$
asidirop@dellpc:/tmp$ uname
Linux
asidirop@dellpc:/tmp$ uname -a
Linux dellpc 2.6.35-32-generic #67-Ubuntu SMP Mon Mar 5 19:39:49 UTC
2012 x86_64 GNU/Linux
asidirop@dellpc:/tmp$ ls -l /bin/sh
lrwxrwxrwx 1 root root 4 2011-02-15 12:15 /bin/sh -> dash
asidirop@dellpc:/tmp$
```

Συμβατότητα κελυφών - προσοχή

29

- Αν στο προηγούμενο παράδειγμα (που είναι γραμμένο για bash), αντί για `#!/bin/bash` στην αρχή βάλουμε `#!/bin/sh` είναι θέμα "**τύχης**" αν θα εκτελεστεί σωστά.
 - στον aetos (debian) θα εκτελεστεί σωστά.
 - στο ubuntu 10.10 με το dash **ΔΕΝ** θα εκτελεστεί σωστά.



```
asidirop@dellpc: /tmp
File Edit View Search Terminal Tabs Help
aetos x asidirop@dellpc: /tmp x
asidirop@dellpc:/tmp$
asidirop@dellpc:/tmp$ head -1 args_test
#!/bin/sh
asidirop@dellpc:/tmp$ ./args_test -t -d
./args_test: 17: 2: not found
You have set the params:
use_command_file=0
use_command_stat=0
file=
asidirop@dellpc:/tmp$
```

Συμβατότητα κελυφών - προσοχή

30

- Συμπέρασμα:
Εάν χρησιμοποιούνται σε ένα script δυνατότητες του bash (πχ: ((, \$((, [[, κτλ) τότε το script πρέπει να ξεκινά οπωσδήποτε με
#!/bin/bash
(και όχι με #!/bin/sh)