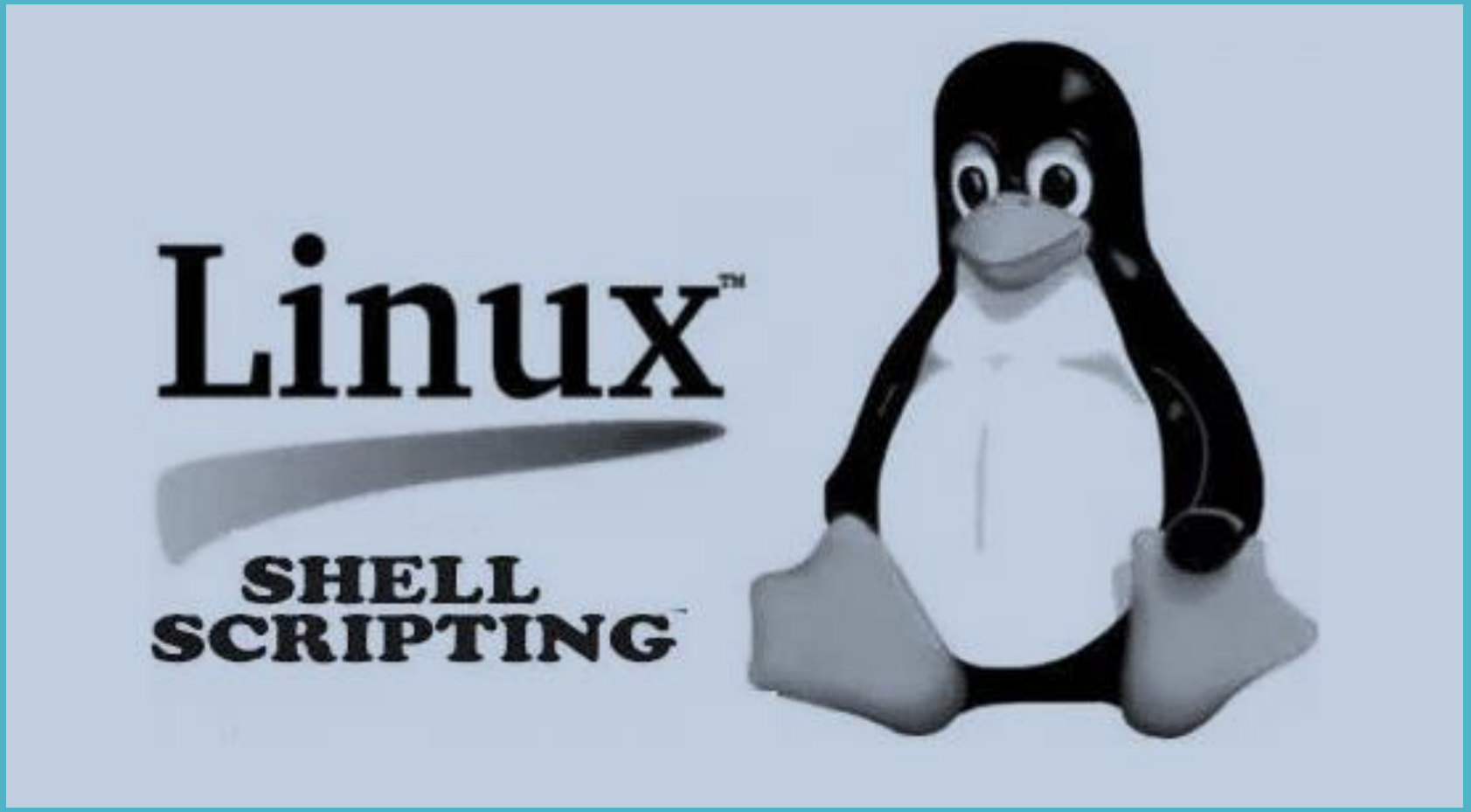
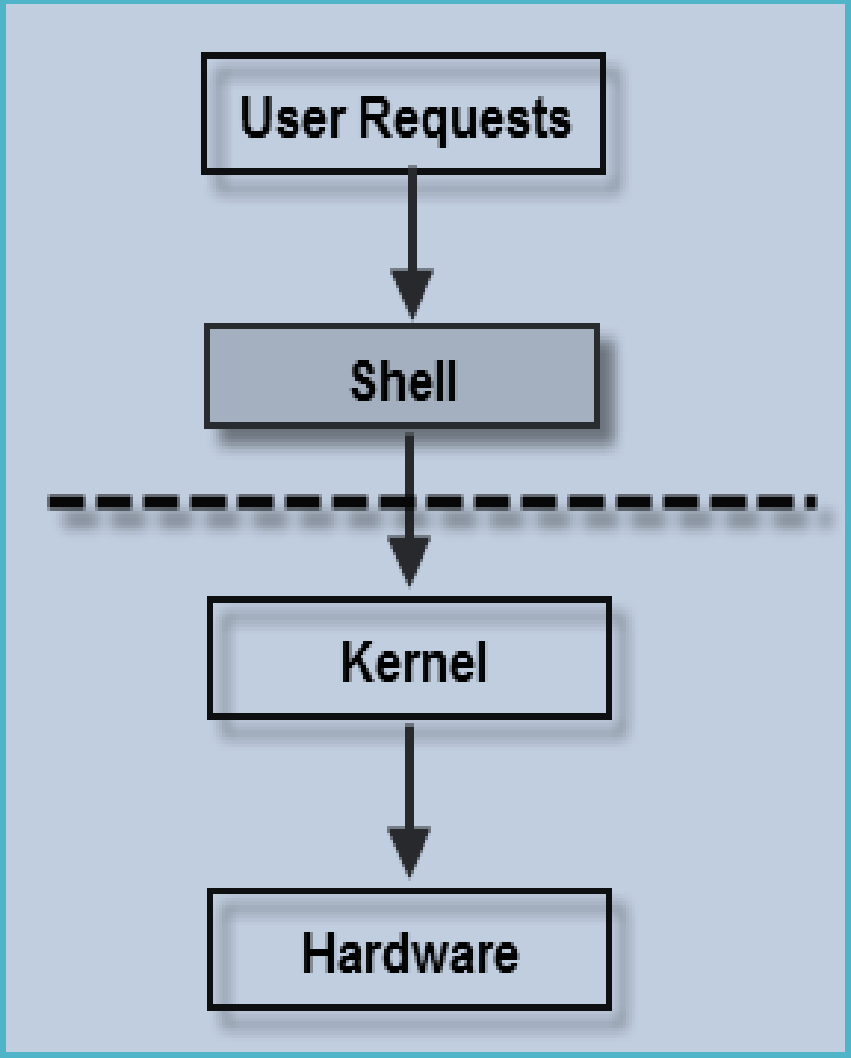
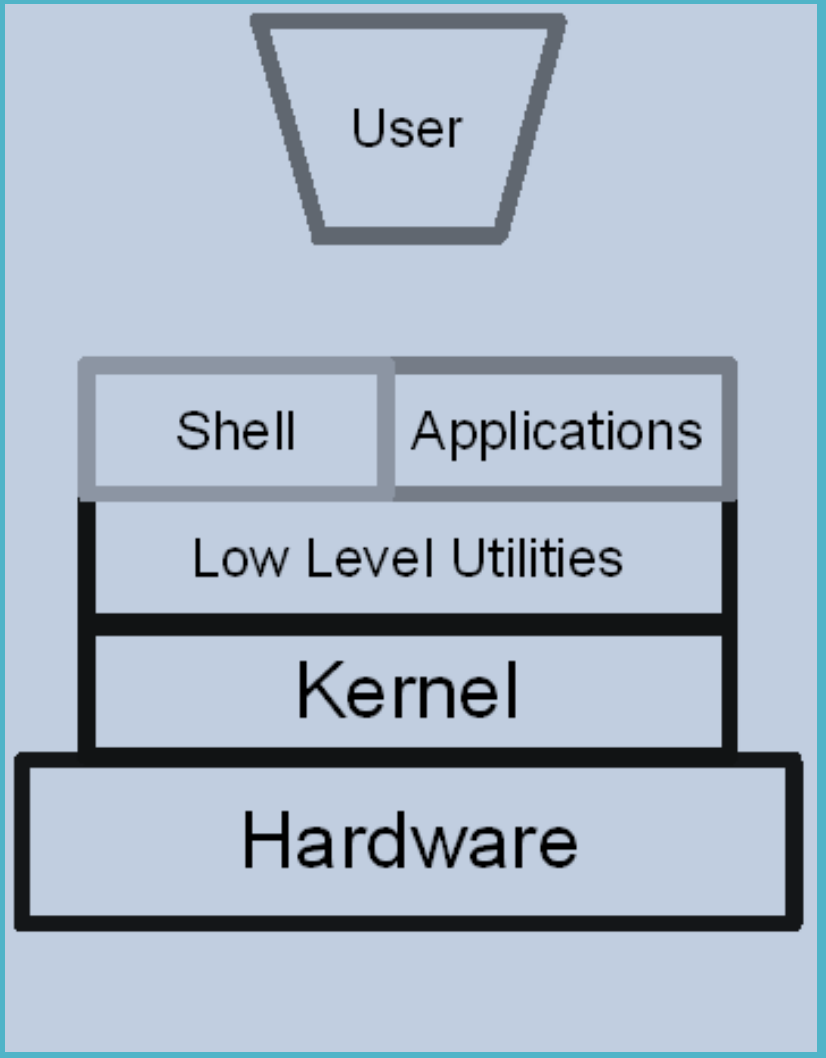
Shell Programming

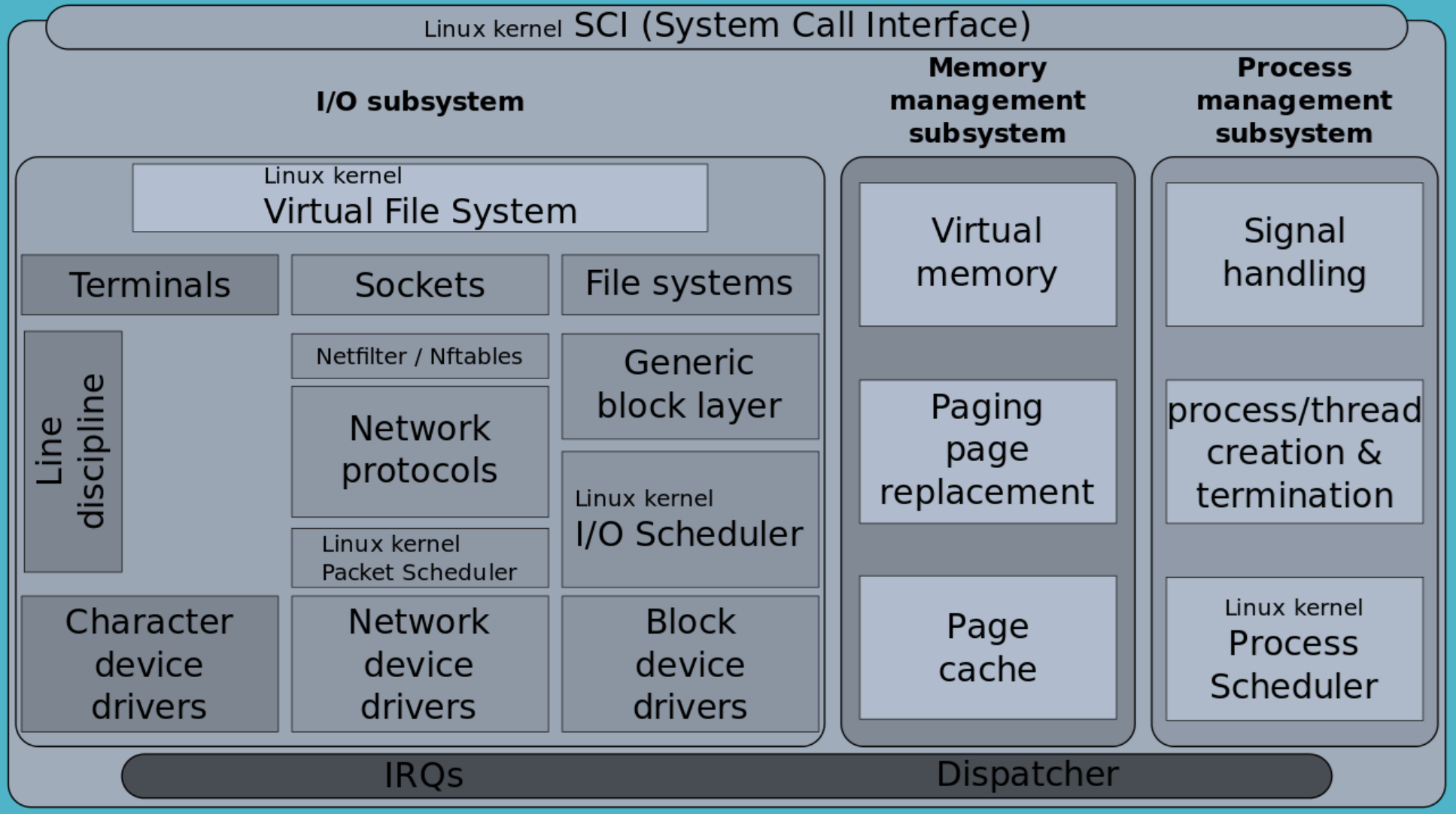


Φλοιός και πυρήνας





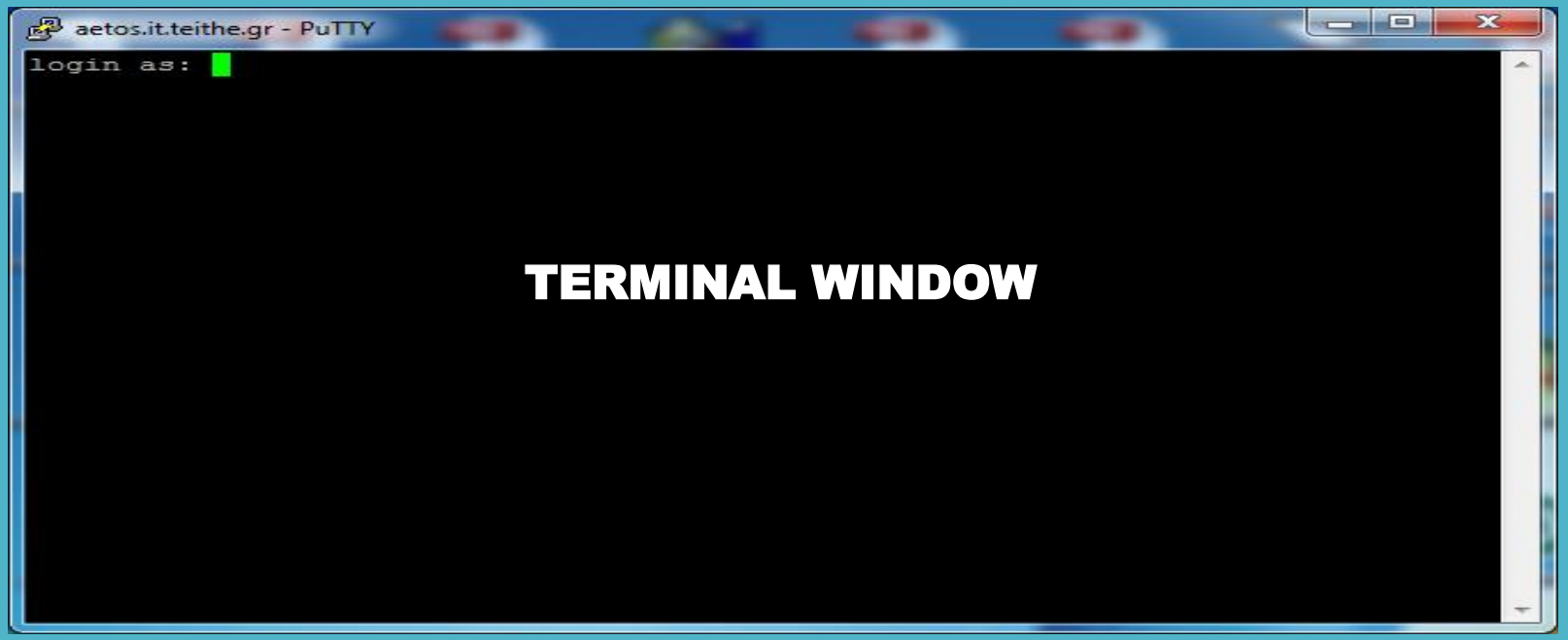
Φλοιός και πυρήνας



Ταυτοποίηση χρήστη

Σε ένα λειτουργικό σύστημα Linux για κάθε χρήστη ορίζονται οι επόμενες ιδιότητες

Login name ή username → συνήθως μέχρι 8 χαρακτήρες.
Password → συνθηματικό εισόδου.
User Id → μοναδικός αριθμός που ταυτοποιεί τον κάθε χρήστη.
Group Id → μοναδικός αριθμός που ταυτοποιεί την πρωτεύουσα ομάδα του χρήστη.
Home Directory → ο προσωπικός κατάλογος του κάθε χρήστη
Φλοιός (Shell) → το περιβάλλον αλληλεπίδρασης του χρήστη με τον πυρήνα



Σύνδεση στο σύστημα

Εάν ο χρήστης καταχωρήσει το σωστό login name και το σωστό password το σύστημα τον μεταφέρει αυτόματα στον προσωπικό του κατάλογο.
Εκτελούνται οι εντολές των αρχείων /etc/profile (για όλους τους χρήστες).

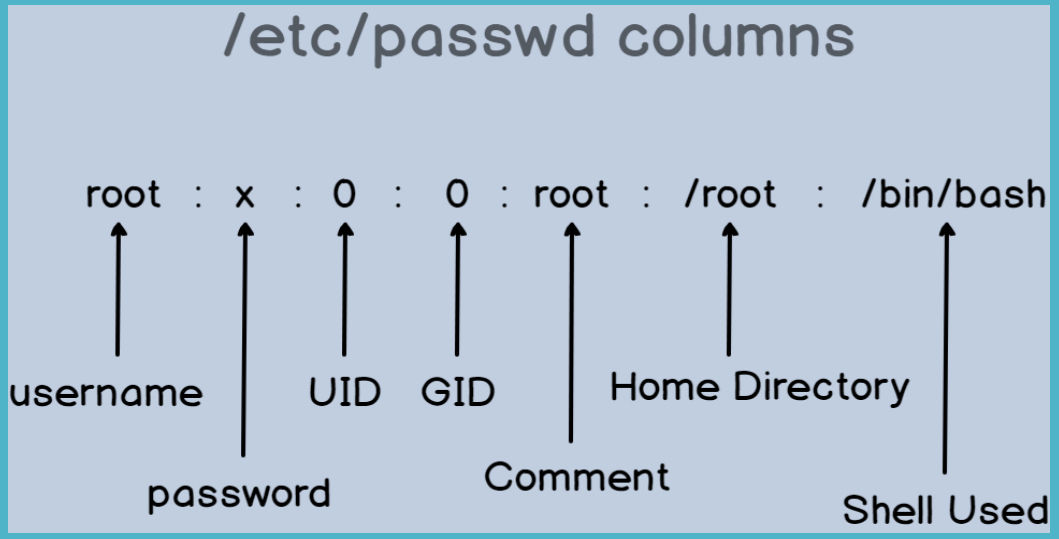
Το home directory κάθε χρήστη μπορεί περιέχει ένα ή περισσότερα από τα ακόλουθα bash startup files, τα οποία περιέχουν εντολές που εφαρμόζονται μόνον για την τρέχουσα χρήση του συστήματος:

~/.bash_profile,
~/.bash_login,
~/.profile,
~/.bashrc, και
~/.bash_logout

```
[root@tecmint ~]# cat /etc/profile
# /etc/profile

# System wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.

pathmunge () {
    case ":${PATH}:" in
        *:"$1":*)
            ;;
        *)
            if [ "$2" = "after" ] ; then
                PATH=$PATH:$1
            else
                PATH=$1:$PATH
            fi
    esac
}


if [ -x /usr/bin/id ]; then
    if [ -z "$EUID" ]; then
        # ksh workaround
        EUID=`id -u`
        UID=`id -ru`
    fi
```

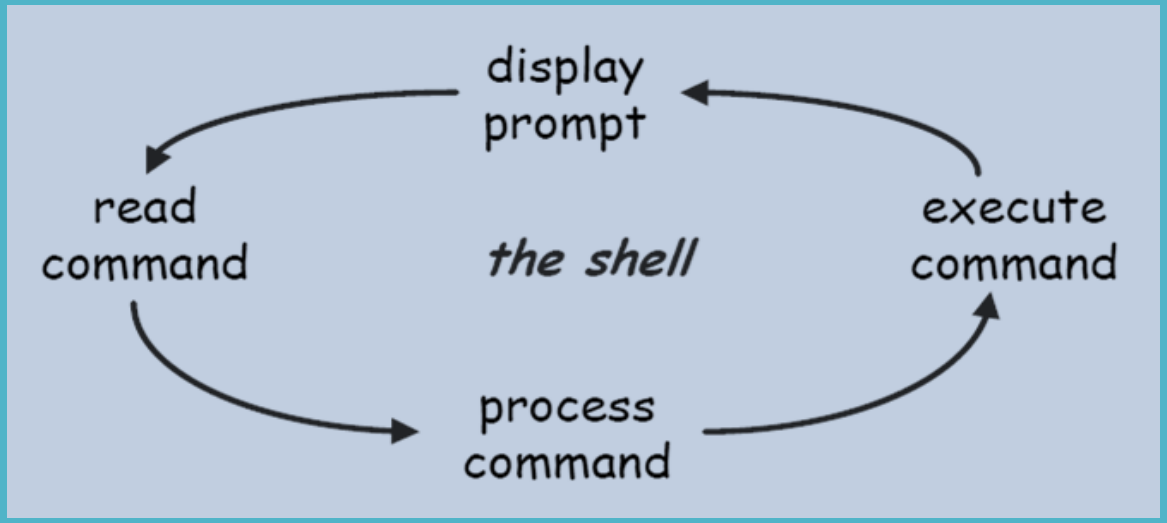
Το αρχείο /etc/passwd

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
```



Η λειτουργία του φλοιού

Ο φλοιός εκτελεί επαναληπτικά τις παρακάτω τέσσερις εργασίες



1. **Εμφανίζει την προτροπή (command prompt)**
2. **Διαβάζει την εντολή του χρήστη**
3. **Αποκωδικοποιεί και επεξεργάζεται την εντολή του χρήστη**
4. **Εκτελεί την εντολή του χρήστη**

Login shell : δημιουργείται κάθε φορά που γίνεται login σε ένα account.

Nonlogin shell : δημιουργείται όταν αρχίζει ένα πρόσθετο bash shell , κατά τη διάρκεια της σύνδεσης του χρήστη, όπως π.χ. όταν ανοίγει ένα terminal window.

Διαθέσιμοι φλοιοί

```
vivek@nixcraft-asus:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/screen
/bin/ksh93
/bin/rksh93
/bin/tcsh
/usr/bin/tcsh
vivek@nixcraft-asus:~$
```

Comparison between shells

- sh (bourne shell)
  - small
  - good scripting capability
  - popular with system administrators
- csh (c-shell)
  - extends sh
  - uses a c-like syntax
  - created by Bill Joy
  - popular with UNIX users
- bash
  - "bourne-again" shell
  - extends sh with some features form csh
  - The linux default
  - We will use bash
- tcsh
  - extension of csh

Ανάκτηση τρέχοντος φλοιού: echo $SHELL

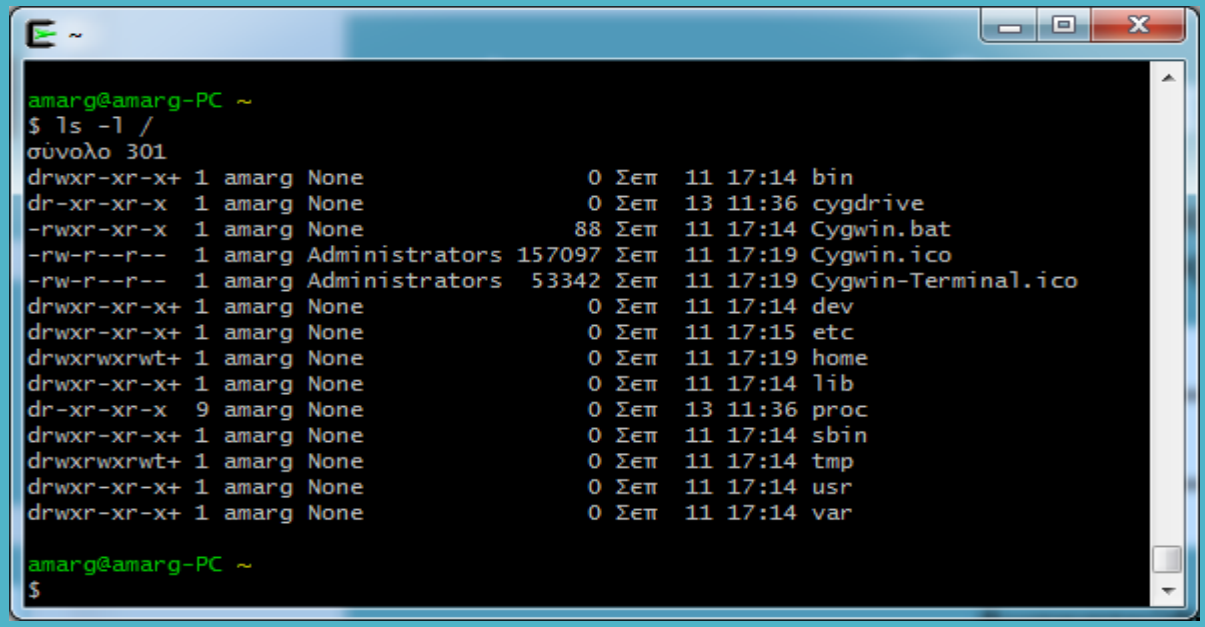
Αλλαγή φλοιού: chsh –s [shell name] π.χ.

chsh –s /bin/tcsh

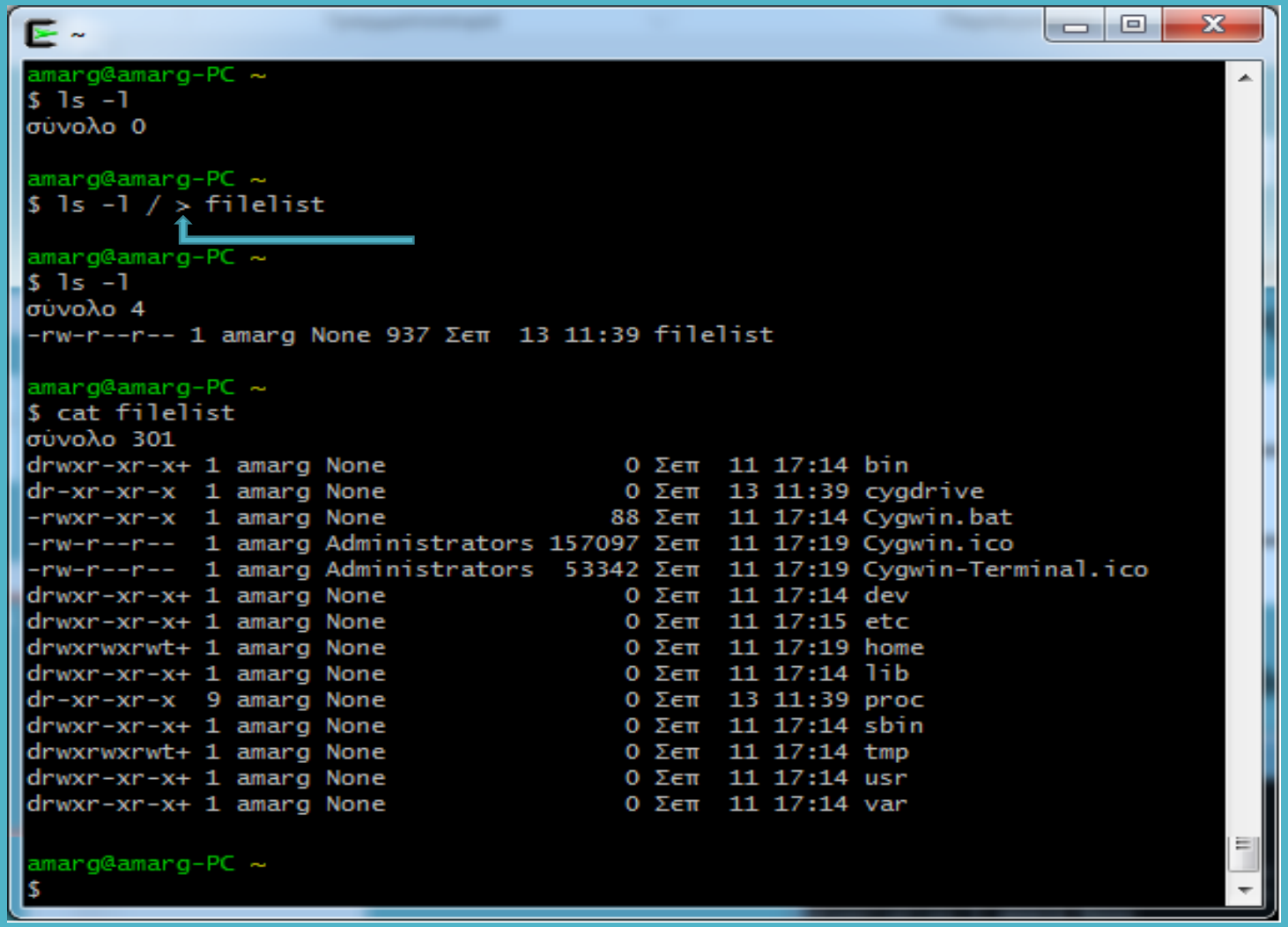
Ανακατεύθυνση και διασωλήνωση

Η προεπιλεγμένη είσοδος των εντολών είναι το πληκτρολόγιο
Η προεπιλεγμένη έξοδος των εντολών είναι η οθόνη

Ανακατεύθυνση εισόδου < **Ανακατεύθυνση εξόδου >**



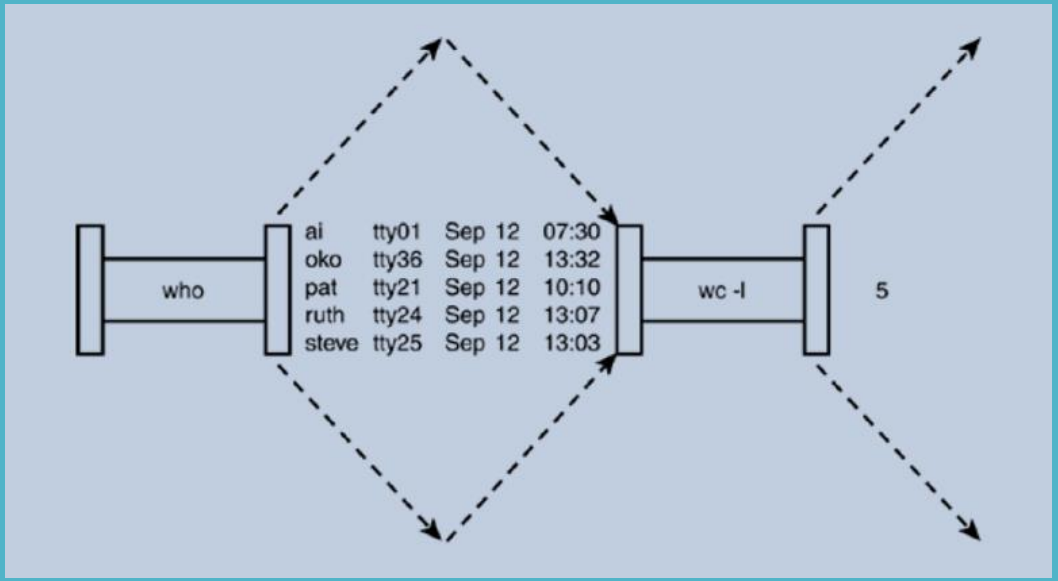
Ανακατεύθυνση και διασωλήνωση



who > connectedUsers
wc –l < connectedUsers

Ανακατεύθυνση και διασωλήνωση

Διασωλήνωση → η έξοδος μιας εντολής δεν εκτυπώνεται στην οθόνη αλλά γίνεται είσοδος μιας άλλης εντολής.



Παραδείγματα

1) **Να μετρηθεί το πλήθος των συνδεδεμένων χρηστών**

who | wc – l

2) **Να μετρηθεί το πλήθος των καταλόγων του συστήματος**

ls -l / | grep -e '^d' | wc -l

3) **Ανάκτηση των ονομάτων χρηστών και αποθήκευσή τους σε αρχείο**

cat /etc/passwd | awk -F : '{print $1}' > alluser.txt

Bash shell → εσωτερικές εντολές

alias
bg
bind
break
builtin
case
cd
command
compgen
complete
continue
declare
dirs
disown
echo
enable
eval
exec
exit
export
fc
fg
getopts
hash
help
history
if
jobs
kill
let
local
logout
popd
printf
pushd
pwd
read
readonly
return
set
shift
shopt
source
suspend
test
times
trap
type
typeset
ulimit
umask
unalias
unset
until
wait
while

Μεταβλητές περιβάλλοντος tsch

- Περιέχουν τις τιμές παραμέτρων που είναι αναγκαίες για τη λειτουργία του συστήματος.
- Αρχικοποιούνται για κάθε χρήστη κατά την είσοδό του στο σύστημα.
- Οι τιμές τους μεταβιβάζονται σε κάθε θυγατρική διεργασία που ξεκινά από το φλοιό.
- Γράφονται με κεφαλαία γράμματα.

- COLUMNS → το πλήθος των στηλών στην οθόνη του τερματικού σύνδεσης
- DISPLAY → χρησιμοποιείται από το γραφικό περιβάλλον
- EDITOR → η διαδρομή προς τον προεπιλεγμένο επεξεργαστή κειμένου
- GROUP → η πρωτεύουσα ομάδα του χρήστη
- HOST → το όνομα του συστήματος στο οποίο χρησιμοποιείται το κέλυφος
- HOSTTYPE → ο τύπος του συστήματος στον οποίο χρησιμοποιείται το κέλυφος
- MACHTYPE → ο τύπος του επεξεργαστή του συστήματος
- OSTYPE → ο τύπος του λειτουργικού συστήματος UNIX
- PATH → λίστα καταλόγων αναζήτησης εκτελέσιμων αρχείων
- REMOTEHOST → το όνομα του συστήματος από το οποίο συνδέθηκε ο χρήστης
- USER → το όνομα του χρήστη που έχει συνδεθεί στο σύστημα.

Μεταβλητές περιβάλλοντος (printenv)

```
PAGER=less
HOSTNAME=icon
MAILCHECK=60
PS1=$
USER=username
MACHTYPE=i486-pc-linux-gnu
EDITOR=emacs
DISPLAY=:0.0
LOGNAME=username
SHELL=/bin/bash
HOSTTYPE=i486
OSTYPE=linux-gnu
HISTSIZE=150
HOME=/home/username
TERM=xterm-debian
TEXEDIT=jed
PATH=/usr/sbin:/usr/sbin:/usr/local/bin:
/usr/bin:/bin:/usr/bin/X11:/usr/games
_=/usr/bin/printenv
```

Τοπικές μεταβλητές κελύφους tsch

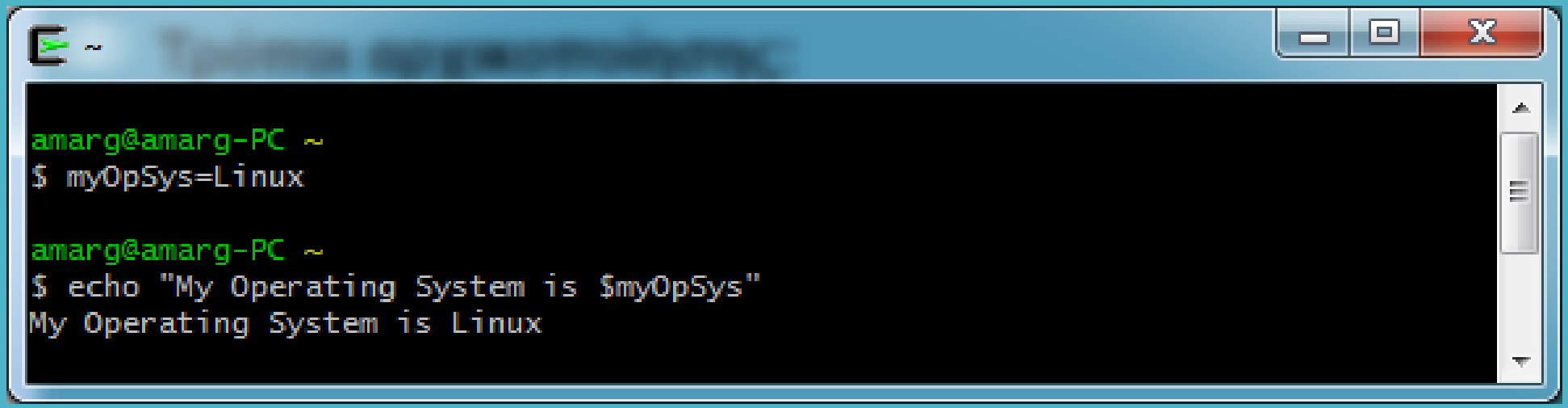
Οι τοπικές μεταβλητές αρχικοποιούνται από το φλοιό
Δεν μεταβιβάζονται σε άλλα προγράμματα που εκτελούνται στο σύστημα.

cdpath → λίστα καταλόγων αναζήτησης υποκαταλόγων για την εντολή cd.
cwd → το πλήρες όνομα της διαδρομής του τρέχοντος καταλόγου.
gid o → ο κωδικός της πρωτεύουσας ομάδας του χρήστη.
group → το όνομα της πρωτεύουσας ομάδας του χρήστη.
home → η απόλυτη διαδρομή προς τον προσωπικό κατάλογο του χρήστη.
inputmode → παίρνει μία από τις τιμές insert και overwrite**.**
path → λίστα καταλόγων αναζήτησης εκτελέσιμων αρχείων.
prompt → ορισμός του command prompt.
uid → ο κωδικός του χρήστη.
user → το όνομα του χρήστη.
shell → η απόλυτη διαδρομή προς το φλοιό που χρησιμοποιείται.

Τοπικές μεταβλητές κελύφους

Τρόποι αρχικοποίησης

1) Απευθείας αρχικοποίηση

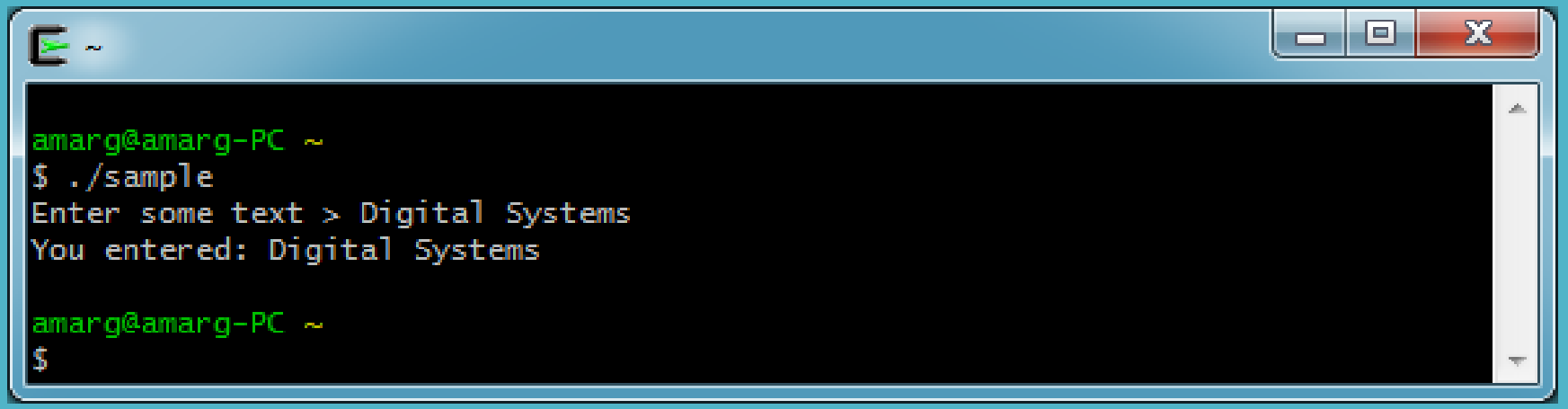


Τοπικές μεταβλητές κελύφους

2) Αρχικοποίηση με την εντολή read

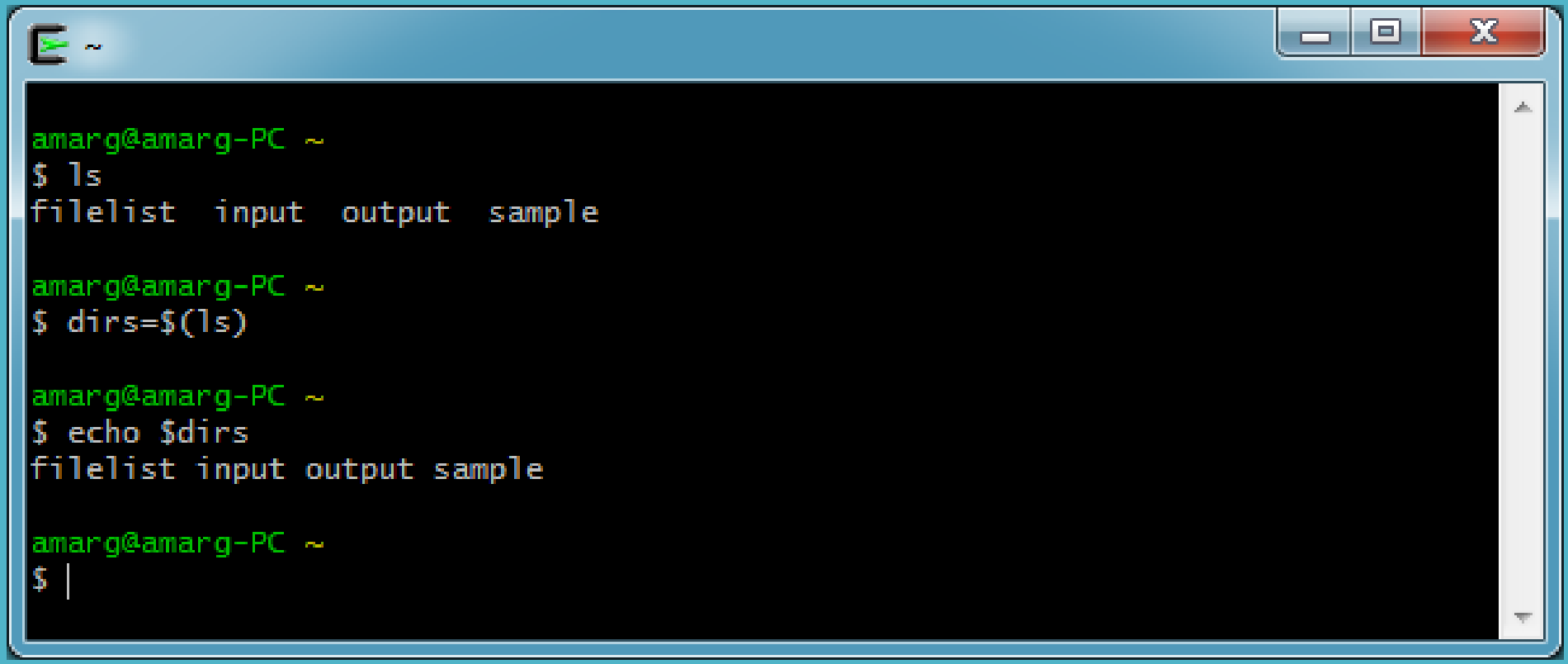
A Sample Script

```
#!/bin/bash

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



Τοπικές μεταβλητές κελύφους

3) Εκχώρηση σε μεταβλητή της εξόδου εντολής



ΓΕΝΙΚΑ ➔ $VARIABLE=$(COMMAND)

Τοπικές μεταβλητές κελύφους

4) Αρχικοποίηση από τη γραμμή εντολών

Η κατάσταση είναι παρόμοια με τη C όπου τα ορίσματα της γραμμής εντολών αποθηκεύονται στα ορίσματα της συνάρτησης main

int main (int argc, char ** argv) ή
int main (int argc, char * argv [])

Έστω πως η main αποτελεί την κύρια συνάρτηση του κώδικα **fileMerge.c**

Εάν στη γραμμή εντολών γράψουμε **fileMerge file1 file2 targetFile**

τότε τα ορίσματα της main αρχικοποιούνται ως

argv [0] = “fileMerge”
argv [1] = “file1”
argv [2] = “file2”
argv [3] = “targeFile”

και **argc = 4**

Τοπικές μεταβλητές κελύφους

Έστω μία εντολή της μορφής

command arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10 arg11 ….

Το κέλυφος περιέχει δέκα μεταβλητές **$0, $1, $2, $3, $4, $5, $6, $7, $8, $9** οι οποίες αρχικοποιούνται ως εξής:

$0 → command ← argv [0]
$1 → arg1 ← argv [1]
$2 → arg2 ← argv [2]
$3 → arg3 ← argv [3]
$4 → arg4 ← argv [4]
$5 → arg5 ← argv [5]
$6 → arg6 ← argv [6]
$7 → arg7 ← argv [7]
$8 → arg8 ← argv [8]
$9 → arg9 ← argv [9]

Παράδειγμα → **find / -name linux –perm 755 –size 100**

$0 $1 $2 $3 $4 $5 $6 $7

Τοπικές μεταβλητές κελύφους

Η προσπέλαση του δέκατου ορίσματος και όσων υπάρχουν από εκει και πέρα γίνεται με επαναληπτική χρήση της **shift**

$0 → command		$0 → arg1		$0 → arg2		$0 → arg3
$1 → arg1		$1 → arg2		$1 → arg3		$1 → arg4
$2 → arg2		$2 → arg3		$2 → arg4		$2 → arg5
$3 → arg3		$3 → arg4		$3 → arg5		$3 → arg6
$4 → arg4	**shift**	$4 → arg5	**shift**	$4 → arg6	**shift**	$4 → arg7
$5 → arg5		$5 → arg6		$5 → arg7		$5 → arg8
$6 → arg6		$6 → arg7		$6 → arg8		$6 → arg9
$7 → arg7		$7 → arg8		$7 → arg9		$7 → arg10
$8 → arg8		$8 → arg9		$8 → arg10		$8 → arg11
$9 → arg9		$9 → arg10		$9 → arg11		$9 → arg12

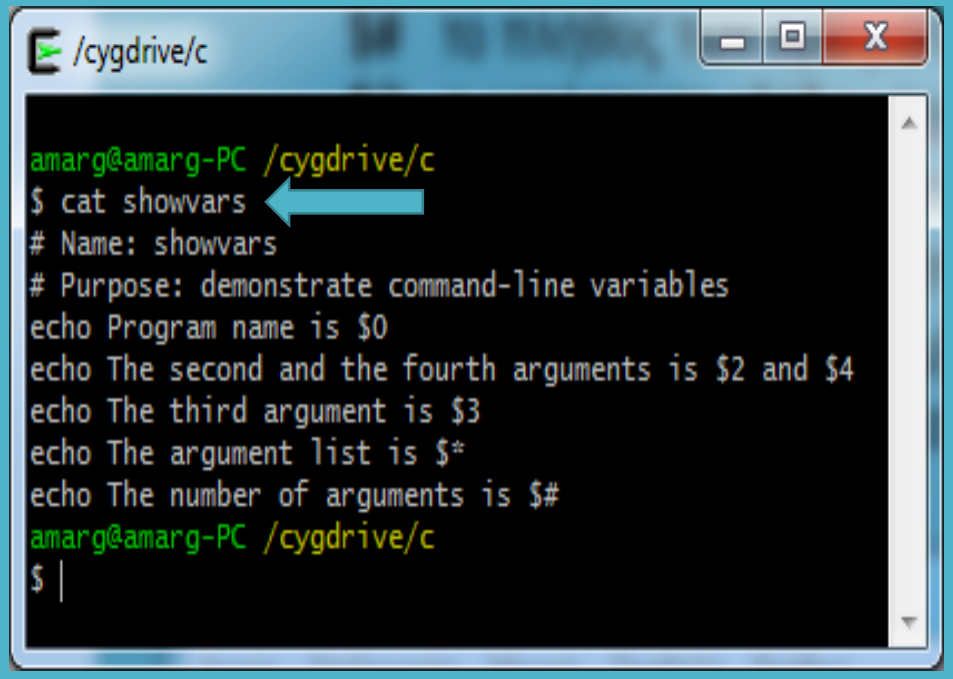
Εναλλακτικά μπορούμε να χρησιμοποιήσουμε άγκιστρα

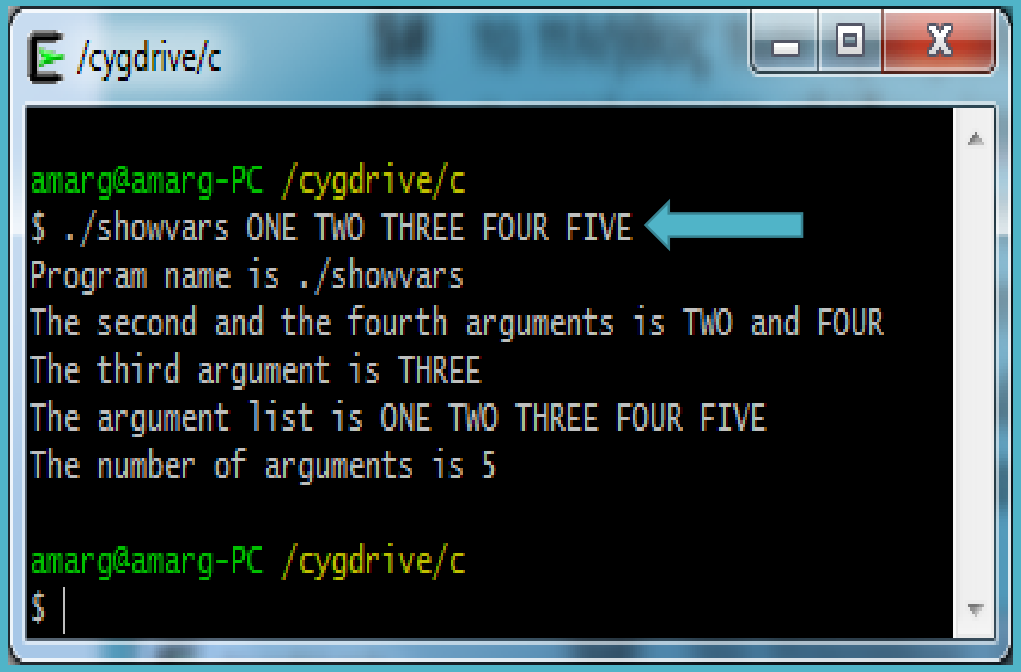
${10}, ${11}, ${12}

Τοπικές μεταβλητές κελύφους

Το σύνολο των μεταβλητών του κελύφους

$# το πλήθος των ορισμάτων θέσης
$? η κατάσταση εξόδου (exit status) της εντολής που εκτελέστηκε τελευταία
$$ ο αριθμός διεργασίας του φλοιού
$! ο αριθμός διεργασίας της διεργασίας που εκτελείται στο παρασκήνιο
$* ένα string που περιλαμβάνει όλα τα ορίσματα
$@ το ίδιο με το $*, εκτός αν χρησιμοποιούνται εισαγωγικά





Shell programming

Τα shell scripts είναι αρχεία κειμένου που περιέχουν κώδικα γραμμένο στη γλώσσα του φλοιού

```
#!/bin/bash
number=0
while [ $number -lt 10 ]; do
      echo "Number = $number"
      number=$((number+1))
done
~
~
~
~
~
~
~
~
"whileExample" 6 lines, 112 characters
```

Για να εκτελεστεί το shell script αρχικά το κάνουμε εκτελέσιμο ως

chmod +x whileExample

και μετά το εκτελούμε ως

./whileExample

Shell programming

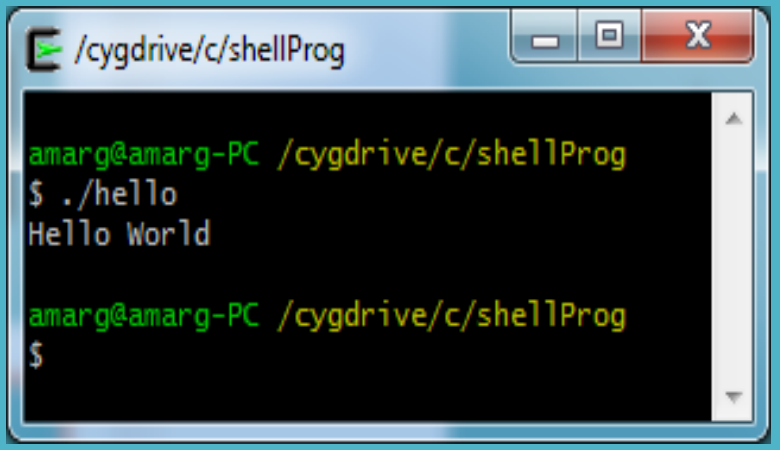
Μερικά απλά shell scripts

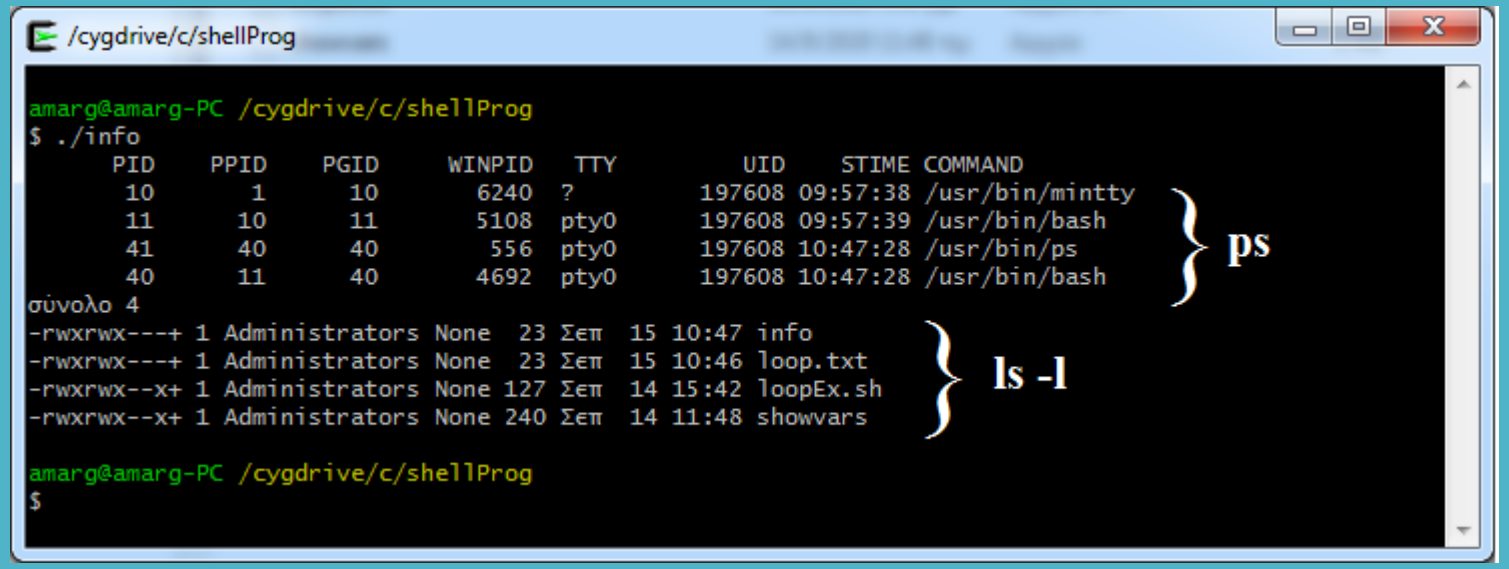
Εκτυπώνει το μήνυμα
Hello world

```
#!/bin/bash
echo "Hello World"
```

Εκτυπώνει την έξοδο των εντολών ps και ls -l

```
#!/bin/bash
ps;  ls -l
```





Shell programming

Τα προγράμματα φλοιού περιέχουν μία ή περισσότερες από τις παρακάτω δομές

- Μεταβλητές
- Λογικές δομές (if, case ...)
- Δομές επανάληψης (while, for, until)
- Συναρτήσεις (functions)
- Σχόλια

Shell programming

Αριθμητικοί και λογικοί τελεστές

+	Ο τελεστής της πρόσθεσης
-	Ο τελεστής της αφαίρεσης
*	Ο τελεστής του πολλαπλασιασμού
/	Ο τελεστής της διαίρεσης
%	Ο τελεστής του υπολοίπου (module) της διαίρεσης
<<	Ο τελεστής της αριστεράς μετατόπισης (left shift)
>>	Ο τελεστής της δεξιάς μετατόπισης (right shift)
<=	Ο τελεστής σύγκρισης «μικρότερο ή ίσο»
>=	Ο τελεστής σύγκρισης «μεγαλύτερο ή ίσο»
<	Ο τελεστής σύγκρισης «μικρότερο»
>	Ο τελεστής σύγκρισης «μεγαλύτερο»
==	Ο τελεστής ισότητας που χρησιμοποιείται στις συγκρίσεις
!=	Ο τελεστής ανισότητας που χρησιμοποιείται στις συγκρίσεις
&	Ο λογικός τελεστής & για πράξεις ανάμεσα σε bits (bitwise AND)
!	Ο λογικός τελεστής OR για πράξεις ανάμεσα σε bits (bitwise OR)
^	Ο λογικός τελεστής XOR για πράξεις ανάμεσα σε bits (bitwise XOR)
&&	Ο λογικός τελεστής AND
!!	Ο λογικός τελεστής OR

Shell programming

Τελεστές καταχώρησης

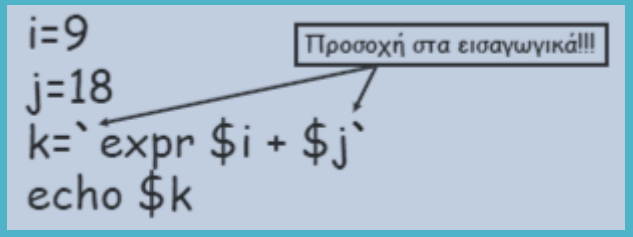
=	Καταχωρεί τιμή σε μεταβλητή του προγράμματος. Για παράδειγμα η πρόταση a=b καταχωρεί στην μεταβλητή a την τιμή της μεταβλητής b
+=	Πραγματοποιεί την πράξη της πρόσθεσης ανάμεσα στις τιμές δύο μεταβλητών και στη συνέχεια καταχωρεί το αποτέλεσμα στην μεταβλητή που βρίσκεται στο αριστερό μέλος της πρότασης. Έτσι η πράξη a+=b είναι ισοδύναμη με την πράξη a=a+b
-=	Πραγματοποιεί την πράξη της αφαίρεσης ανάμεσα στις τιμές δύο μεταβλητών και στη συνέχεια καταχωρεί το αποτέλεσμα στην μεταβλητή που βρίσκεται στο αριστερό μέλος της πρότασης. Έτσι η πράξη a-=b είναι ισοδύναμη με την πράξη a=a-b
*=	Πραγματοποιεί την πράξη του πολλαπλασιασμού ανάμεσα στις τιμές δύο μεταβλητών και στη συνέχεια καταχωρεί το αποτέλεσμα στην μεταβλητή που βρίσκεται στο αριστερό μέλος της πρότασης. Έτσι η πράξη a*=b είναι ισοδύναμη με την πράξη a=a*b
/=	Πραγματοποιεί την πράξη της διαίρεσης ανάμεσα στις τιμές δύο μεταβλητών και στη συνέχεια καταχωρεί το αποτέλεσμα στην μεταβλητή που βρίσκεται στο αριστερό μέλος της πρότασης. Έτσι η πράξη a/=b είναι ισοδύναμη με την πράξη a=a/b
%=	Πραγματοποιεί την πράξη της διαίρεσης ανάμεσα στις τιμές δύο μεταβλητών και στη συνέχεια καταχωρεί το υπόλοιπο της διαίρεσης στην μεταβλητή που βρίσκεται στο αριστερό μέλος της πρότασης. Έτσι η πράξη a%=b είναι ισοδύναμη με την πράξη a=a%b
<<=	Πραγματοποιεί την πράξη της αριστεράς μετατόπισης ανάμεσα στις τιμές δύο μεταβλητών και στη συνέχεια καταχωρεί το αποτέλεσμα στην μεταβλητή που βρίσκεται στο αριστερό μέλος της πρότασης. Έτσι η πράξη a<<=b είναι ισοδύναμη με την πράξη a=a<<b
>>=	Πραγματοποιεί την πράξη της δεξιάς μετατόπισης ανάμεσα στις τιμές δύο μεταβλητών και στη συνέχεια καταχωρεί το αποτέλεσμα στην μεταβλητή που βρίσκεται στο αριστερό μέλος της πρότασης. Έτσι η πράξη a>>=b είναι ισοδύναμη με την πράξη a=a>>b

Shell programming

Αριθμητικές πράξεις

```
#!/bin/bash

first_num=0
second_num=0

echo -n "Enter the first number --> "
read first_num
echo -n "Enter the second number -> "
read second_num

echo "first number + second number = $((first_num + second_num))"
echo "first number - second number = $((first_num - second_num))"
echo "first number * second number = $((first_num * second_num))"
echo "first number / second number = $((first_num / second_num))"
echo "first number % second number = $((first_num % second_num))"
echo "first number raised to the"
echo "power of the second number   = $((first_num ** second_num))"
```

Χρησιμοποιώντας την expr



Shell programming

Αριθμητικές πράξεις

```
~$ ./calc
Enter the first number --> 100
Enter the second number -> 5
first number + second number = 105
first number - second number = 95
first number * second number = 500
first number / second number = 20
first number % second number = 0
first number raised to the
power of the second number = 10000000000
~$
```

Εντολές διακλάδωσης

Δομή if – elif - fi

```
if expression;
   then
     commands;
   elif expression;
     then
     commands;
   elif expression;
     then
     commands  ...
   else
     commands;
fi
```

Numeric Comparison	Returns true (0) if:
[$num1 -eq $num2]	num1 equals num2
[$num1 -ne $num2]	num1 does not equal num2
[$num1 -lt $num2]	num1 is less than num2
[$num1 -gt $num2]	num1 is greater than num2
[$num1 -le $num2]	num1 is less than or equal to num2
[$num1 -ge $num2]	num1 is greater than or equal to num2

String Comparison	Returns true (0) if:
[str1 = str2]	str1 equals str2
[str1 != str2]	str1 does not equal str2
[str1 < str2]	str1 precedes str2 in lexical order
[str1 > str2]	str1 follows str2 in lexical order
[-z str1]	str1 has length zero (holds null value)
[-nstr1]	str1has nonzero length (contains one or more characters)

Εντολές διακλάδωσης

Παραδείγματα

```
~$ cat testIf
#!/bin/bash
echo -n "Enter a number --> "
read number
if [ $number -eq 100 ]
then
 echo "Number is equal to 100"
elif [ $number -lt 100 ]
then
 echo "Number is less than 100"
else
 echo "Number is greater than 100"
fi
~$
```

```
~$ ./testIf
Enter a number --> 200
Number is greater than 100
~$ ./testIf
Enter a number --> 100
Number is equal to 100
~$ ./testIf
Enter a number --> 35
Number is less than 100
~$
```

```
~$ cat testIf
#!/bin/bash
echo -n "Enter a number --> "
read number
if [ $number -gt 100 ] && [ $number -lt 200 ]
then
 echo "The number lies between 100 and 200"
else
 echo "Number lies outside the interval [100, 200]"
fi
~$
```

```
~$ ./testIf
Enter a number --> 150
The number lies between 100 and 200
~$ ./testIf
Enter a number --> 250
Number lies outside the interval [100, 200]
~$
```

Σύνθετες εντολές με τους λογικούς τελεστές && και ||

Εντολές διακλάδωσης

Παραδείγματα

Έλεγχος άρτιας ή περιττής τιμής

```
#!/bin/bash
number=0

echo -n "Enter a number > "
read number

echo "Number is $number"
if [ $((number % 2)) -eq 0 ]; then
    echo "Number is even"
else
    echo "Number is odd"
fi
```

Η εντολή case

Βασική σύνταξη

```
case expression in
    pattern1 )
        statements ;;
    pattern2 )
        statements ;;
    ...
esac
```

Πρόγραμμα

```
~$ cat testCase
#!/bin/sh

# take a number from user
echo "Enter number:"
read num
case $num in
  1)
    echo "It's one!"
    ;;
  2)
    echo "It's two!"
    ;;
  3)
    echo "It's three!"
    ;;
  *)
    echo "It's something else!"
    ;;
esac
echo "End of script."
~$
```

Έξοδος

```
~$ ./testCase
Enter number:
1
It's one!
End of script.
~$ ./testCase
Enter number:
2
It's two!
End of script.
~$ ./testCase
Enter number:
3
It's three!
End of script.
~$ ./testCase
Enter number:
4
It's something else!
End of script.
~$
```

Βρόχοι επανάληψης (for)

Βρόχος for

for i in λίστα λέξεων;
do
εντολές
done

```
#!/bin/bash
for i in January February March April May June; do
echo $i
done
~
```

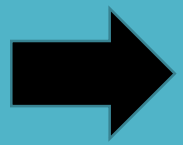
(το $i παίρνει διαδοχικά τις τιμές της λίστας λέξεων)

```
~$ ./months
January
February
March
April
May
June
~$
```

Βρόχοι επανάληψης (for)

Η λίστα των λέξεων στο βρόχο for μπορεί να προκύπτει από την εκτέλεση εντολής

```
~$ cat ./userNames
#!/bin/bash
for i in $(cat /etc/passwd | awk -F : '{print $1}'); do
echo $i
done
```



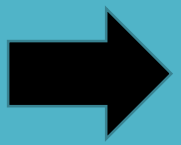
```
~$ ./userNames
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
_apt
systemd-timesync
systemd-network
systemd-resolve
messagebus
salvus
ntp
sshd
rtkit
cups-pk-helper
usbmux
dnsmasq
avahi
geoclue
pulse
saned
colord
gdm
epmd
```

Υπολογισμός μεγέθους καταλόγου

```
~$ ls -l
total 5
-rw-r--r-- 1 user user   0 Sep 15 08:41 'Welcome to CoCalc.term'
-rwxr-xr-x 1 user user 573 Sep 15 08:25  calc
-rwxr-xr-x 1 user user  76 Sep 15 08:32  months
-rwxr-xr-x 1 user user 119 Sep 15 09:09  totalSize
-rwxr-xr-x 1 user user  81 Sep 15 08:46  userNames
~$
```

Υπολογισμός μεγέθους καταλόγου

```
~$ cat ./totalSize
#!/bin/bash
size=0
for i in $(ls -l | awk '{print $5}'); do
        echo $i
        size=$((size+i))
done
echo "Total size is $size"
~$
```

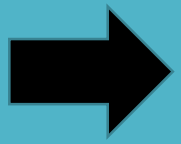


```
~$ ./totalSize
0
573
76
119
81
Total size is 849
~$
```

Βρόχοι επανάληψης (while)

```
while [ condition ]
do
        command1
        command2
        ..
        ....
        commandN
done
```

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



```
~$ ./testWhile
Number = 0
Number = 1
Number = 2
Number = 3
Number = 4
Number = 5
Number = 6
Number = 7
Number = 8
Number = 9
~$
```

Βρόχοι επανάληψης (while)

Internal Field Separator (IFS) ➔ :
Read (word segmentation of strings)

```
~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
```

```
~$ cat readPasswd
#!/bin/bash
file=/etc/passwd
# set field delimiter to :
# read all 7 fields into 7 vars
while IFS=: read -r user enpass uid gid desc home shell
do
    # only display if UID >= 500
        [ $uid -ge 500 ] && echo "User $user ($uid) assigned \"$home\" home directory with $shell shell."
done < "$file"
~$
```

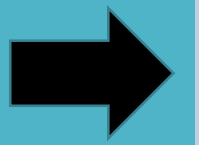
while there are data to read (return code is zero)

```
~$ ./readPasswd
User nobody (65534) assigned "/nonexistent" home directory with /usr/sbin/nologin shell.
User salvus (1000) assigned "/home/salvus" home directory with /bin/bash shell.
User user (2001) assigned "/home/user" home directory with /bin/bash shell.
User sbt (999) assigned "/home/sbt" home directory with /bin/false shell.
User rstudio-server (998) assigned "/home/rstudio-server" home directory with /bin/sh shell.
~$
```

Βρόχοι επανάληψης (until)

```
until [ condition ]
do
    command1
    command2
    ...
    ....
    commandN
done
```

```
~$ cat ./testUntil
#!/bin/bash
number=0
until [ $number -ge 10 ]; do
echo "Number = $number"
number=$((number + 1))
done
~$
```



```
~$ ./testUntil
Number = 0
Number = 1
Number = 2
Number = 3
Number = 4
Number = 5
Number = 6
Number = 7
Number = 8
Number = 9
~$
```

Η εντολή test

Expression	*Description*
-d *file*	True if *file* is a directory.
-e *file*	True if *file* exists.
-f *file*	True if *file* exists and is a regular file.
-L *file*	True if *file* is a symbolic link.
-r *file*	True if *file* is a file readable by you.
-w *file*	True if *file* is a file writable by you.
-x *file*	True if *file* is a file executable by you.
file1 -nt *file2*	True if *file1* is newer than (according to modification time) *file2*
file1 -ot *file2*	True if *file1* is older than *file2*
-z *string*	True if *string* is empty.
-n *string*	True if *string* is not empty.
string1 = *string2*	True if *string1* equals *string2.*
string1 != *string2*	True if *string1* does not equal *string2.*

Η εντολή test

```
#!/bin/bash
for filename in $@;
do
    if [ -f $filename ]; then
       result="$filename is a regular file"
    else
       if [ -d $filename ]; then
          result="$filename is a directory"
       fi
    fi
    if [ -w $filename ]; then
       result="$result and it is writable"
    else
       result="$result and it is not writable"
    fi
    echo "$result"
done
```

Παραδείγματα

Εκτύπωση του μήκους της κάθε λέξης ενός αρχείου κειμένου

```
#!/bin/bash
count=0
for i in $(cat testfile);
do
    count=$((count + 1))
    echo "Word $count ($i) contains $(echo -
  n $i | wc -c) characters"
done
```

Παραδείγματα

Εύρεση του μέσου όρου μιας ακολουθίας αριθμών που καταχωρούνται από το πληκτρολόγιο

```
#!/bin/bash
sum=0
num=0
while true; do
echo "-n enter a number [0-100] (0 for quit) :";read score
if [ $score -lt 0 ] || [ $score -gt 100 ]; then
    echo "try again!!"
elif [ $score -eq 0 ]; then
    echo "average = $average"
    exit 0
else
    sum=$((sum+score))
    num=$((num+1))
    average=$((sum/num))
fi
done
echo "exit - end"
```

Παραδείγματα

Να γραφεί ένα shell script που θα δέχεται το login name ενός χρήστη και θα εμφανίζει πόσες φορές έχει κάνει logged on (χρήση των εντολών who, grep, wc)

ΛΥΣΗ

```
#!/bin/bash
times=`who | grep $1 | wc -l`
echo "$1 is logged on $times times"
```