ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

Λειτουργικά Συστήματα

Linux & Shell Programming

Linux

Τι είναι το Linux

- An Operating System, created under the model of **free and open-source software development and distribution**

- Υπάρχουν πολλές εκδόσεις (distributions) του Linux

- Όλες τους μοιράζονται τον ίδιο Linux kernel.

Major players

Linux Torvalds

- Wrote initial Linux OS
- Contributed non-stop since 1991
- Now more than 10,000 developers and major companies develop it

Richard Stallman

- Founded GNU
  - "**G**NU is **N**ot **U**nix"
- Free software, tools, libraries

Linux in the real world

- World Wide Web
  - 77-90% of all web servers
  - 96% of 1 million top web sites
- Supercomputers
  - 100% of top 500 fastest supercomputers
- Cloud & Enterprise
  - 90% of public cloud workloads
- Smartphones
  - Android
  - iOS (BSD → Darwin → iOS)

Why Linux

- Free and open-source
- Powerful for
  - Research data centers
  - Desktops / laptops
  - Smartphones
- Universal
- Community driven
  - with significant business support

The Bash Shell

Resources

Μπορείτε να βρείτε ένα καταπληκτικό tutorial για το bash shell στο:

http://linuxcommand.org/

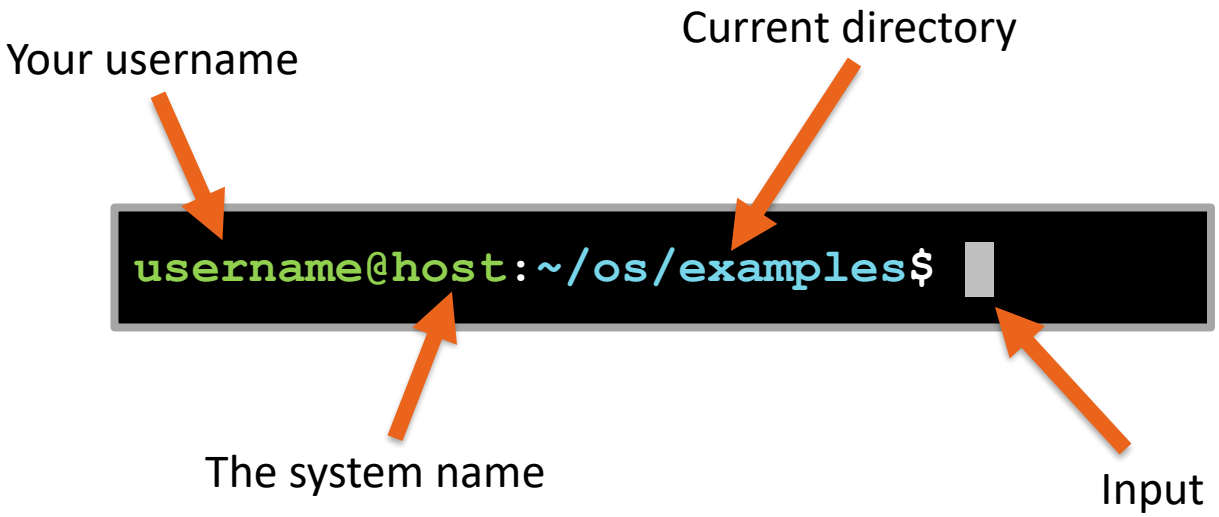
Γενικά

- Το κέλυφος (shell) προσφέρει ένα εναλλακτικό περιβάλλον από την κονσόλα
  - Επιτρέπει τον συνδυασμό εντολών με τη χρήση script
  - Προσφέρει εναλλακτικούς τρόπους για την επίτευξη σύνθετων ενεργειών
  - Επιτρέπει αποθήκευση μεταβλητών
- Υπάρχουν πολλά διαφορετικά κελύφη korn, tcsh, zsh . . . Κάθε χειριστής έχει ένα προεπιλεγμένο κέλυφος
  - Η επιλογή διατηρείται στο αρχείο */etc/passwd*
    spyros:x:1000:1000:,,,:/home/spyros:/bin/bash
  - Η εντολή *chsh* αλλάζει το κέλυφος
- Προσφέρει αρχεία ρυθμίσεων διαφορετικά για κάθε λογαριασμό

Linux: The Shell

- Program running in a terminal
  - Interprets commands and sends them to the OS
- Provides:
  - Built-in commands
  - Programming control structures
  - Environment variables
- Linux supports multiple shells
  - sh / csh / tcsh / korn / bash / zsh / …
  - The default shell is **bash** (Bourne-Again Shell)

The shell prompt



Το σύμβολο ~ (tilda ή περισπωμένη) είναι συντομογραφία για το home directory του χρήστη

Some commands

- Some commands
  - whoami # my login name
  - Hostname # name of this computer
  - echo “Hello, world” # print something on the screen
  - echo $HOME # print environment variable
  - echo my login is $(whoami) # replace $(xx) with program output
  - date # print current date / time
  - cal # print current month’s calendar
- Manual pages (help)
  - man <command> # display manual page for <command>
  - <command> -h/--help # *usually* displays short help

Commands

- "Small programs that do one thing well"

- The Unix Programming Environment, Kernighan and Pike:

  "… at its heart is the idea that the power of a system comes more from the ***relationships among programs*** *than from the programs themselves. Many UNIX programs do quite trivial things in isolation, but, combined with other programs, become general and useful tools."*

Some of the text processing utilities

- awk — Pattern scanning and processing language
- cat — Display file(s)
- cut — Extract selected fields of each line of a file
- diff — Compare two files
- grep — Search text for a pattern
- head — Display the first part of files
- less — Display files on a page-by-page basis
- od — Dump files in various formats
- sed — Stream editor (esp. search and replace)
- sort — Sort text files
- split — Split files
- tail — Display the last part of a file
- tr — Translate/delete characters
- uniq — Filter out repeated lines in a file
- wc — Line, word and character count
- tar — File archive (similar to zip)

Ενσωματωμένες Εντολές – Builtin Commands

Εντολή	Περιγραφή	Παράδειγμα
cd	Αλλαγή φακέλου	**cd ..**
declare	Ορισμός μεταβλητής	**declare myvar**
echo	Εμφάνιση κειμένου στη βασική έξοδο (stdout)	**echo hello**
exec	Αντικατάσταση του bash από μια άλλη διεργασία	**exec ls**
exit	Τερματισμός του bash	**exit**
export	Ορισμός καθολικής μεταβλητής	**export myvar=1**
history	Εμφάνιση ιστορικού εντολών	**history**
kill	Αποστολή σήματος σε μια διεργασία	**kill 1121**
let	Υπολογισμός μιας αριθμητικής πράξης	**let myvar=3+5**

Ενσωματωμένες Εντολές – Builtin Commands

Εντολή	Περιγραφή	Παράδειγμα
`local`	Ορισμός τοπικής μεταβλητής	`local myvar=5`
`pwd`	Εμφάνιση τρέχοντος φακέλου (print working directory)	`pwd`
`read`	Ανάγνωση από τη βασική είσοδο (stdin) σε μια μεταβλητή	`read myvar`
`readonly`	Κλειδώνει μια μεταβλητή	`readonly myvar`
`return`	Ολοκλήρωση μιας συνάρτησης και επιστροφή τιμής	`return 1`
`set`	Εμφάνιση μεταβλητών	`set`
`shift`	Μεταθέτει τις παραμέτρους	`shift 2`
`test`	Έλεγχος μιας έκφρασης	`test –d temp`
`trap`	Παρακολούθηση ενός σήματος	`trap "echo signal" 3`

I/O redirection

- Οι εντολές παράγουν έξοδο – χρησιμοποιούμε το επίθεμα **>** για την προώθηση (**stdout redirection**) σε κάποιο αρχείο
  # ls > filelist
  - Θα δημιουργηθεί ένα νέο αρχείο με όνομα filelist
  - Αν υπάρχει ήδη, το νέο αρχείο θα αντικαταστήσει το παλιό (**overwrite**)
  - Χρησιμοποιούμε το επίθεμα **>>** για την προώθηση σε κάποιο υπάρχον αρχείο (**append**)
    - # ls -lt /root/doc >> /root/filelist

- Οι εντολές απαιτούν είσοδο – χρησιμοποιούμε το επίθεμα **<** για την προώθηση ενός αρχείου ως είσοδο (**stdin redirection**)
  # sort < /root/filelist

- Για να προωθήσουμε την έξοδο μιας εντολής στην είσοδο μιας άλλης – χρησιμοποιούμε το επίθεμα **|** (**pipe**)

```
# who | sort                  #ταξινόμηση καταλόγου χειριστών
# ls /root | grep rc | wc -l  # καταμέτρηση αρχείων με filename που
                              #περιέχει το substring 'rc'
```

Διεργασίες

- Μπορούμε να εκτελέσουμε εντολές **σειριακά** διαχωρίζοντας τις εντολές με **;**
  - Εκτελούνται όλες οι εντολές και **όταν ολοκληρωθεί και η τελευταία**, προσφέρεται νέο prompt

    # who | sort ; date

- Μπορούμε να εκτελέσουμε εντολές **παράλληλα** διαχωρίζοντάς τες με **&**
  - Εκτελούνται όλες οι εντολές και **προσφέρεται άμεσα** νέο prompt

    # pr junk | lpr &

- Η εκτέλεση μια εντολής ειναι μια *διεργασία*
  - Η εντολή *ps* εμφανίζει τις τρέχουσες διεργασίες
  - Η εντολή *wait* περιμένει μέχρι να ολοκληρωθούν όλες οι εντολές που εκτελέστηκαν με &

Κατάλογος διεργασιών

```
# ps -a
PID    TTY    TIME   CMD
106    c1     0:01   -sh
4114   co     0:00   /bin/sh /usr/bin/packman
2114   co     0:00   -sh
6762   c1     0:00   ps -a
87     c2     0:00   getty
90     c3     0:00   getty
```

Παράμετρος **a** -- εμφάνιση διεργασιών που δημιουργήθηκαν από κονσόλες
Στήλη **PID** -- μοναδική ταυτότητα διεργασίας
Στήλη **TTY** -- κονσόλα που δημιούργησε την διεργασία
Στήλη **TIME** -- συνολικός χρόνος εκτέλεσης
Στήλη **CMD** -- εντολή που εκτελέστηκε

Εργαλεία διαχείρισης διεργασιών

- Τερματισμός διεργασίας – εντολή *kill [PID]*
- Μπορούμε να εκτελέσουμε μια εντολή με διαφορετική προτεραιότητα
  - πρόθεμα **nice**
    # nice du| sort -n &
- Μπορούμε να καθυστερήσουμε την εκτέλεση μιας εντολής
  - πρόθεμα **at**

```
# at 1500
ls -l / /root /dir | wc > allfiles
date > lpr-starttime
pr allfiles | lpr ; date > lpr-endtime
^D
at: /usr/spool/at/07.111.1500.67 created
#
```

Εντολή echo (1)

```
bash-3.00# echo hello there
hello there
bash-3.00# let myvar=1; echo $myvar
1
bash-3.00# echo *
junk lpr-starttime temp
bash-3.00# echo print '*' "don't"
print * don't
```

- Βασικός τρόπος για τη δημιουργία εξόδου
- Εκτυπώνει τις τιμές των μεταβλητών
- Αναγνωρίζει κάποιους ειδικούς χαρακτήρες (ή μετα-χαρακτήρες)

Εντολή echo (2)

- Μπορεί να περιέχει περισσότερες από μία γραμμές

```
bash-3.00# echo ’hello
there’
hello
there
bash-3.00# echo hello\
    there
hello there
bash-3.00# echo ‘date‘
Mon Apr 30 16:12:21 GMT 2007
bash-3.00# echo -n ‘date‘ " "
Mon Apr 30 16:12:21 GMT 2007 bash-3.00#
```

Wildcards

- Το BASH χρησιμοποιεί κάποιους ειδικούς χαρακτήρες ως wildcards (μπαλαντέρ)
  - **?** : ένας (ακριβώς ένας!) χαρακτήρας
    - `$ ls /etc/rc.????`
  - ***** : από μηδέν ως πολλοί χαρακτήρες
    - `$ ls /etc/rc.*`
  - **[...]** : συγκεκριμένοι χαρακτήρες,
    - `$ ls [abc]oo.c`

      `aoo.c, boo.c, coo.c`

- Μπορούμε να τους χρησιμοποιήσουμε σε συνδυασμό με όλες τις εντολές

- Ερώτηση: τι κάνει η παρακάτω εντολή;
  - mv *.x *.y
  - Hint: Για να δείτε *τι* θα κάνει, αντικαταστήστε το mv με echo, για να δείτε πως αντικαθιστά τα arguments το BASH shell.

Μεταβλητές Περιβάλλοντος

- Το κέλυφος επιτρέπει τον ορισμό μεταβλητών
  - Είναι case-sensitive, δηλ. $VAR != $var
- Οι αρχικές τιμές των μεταβλητών ορίζονται στο αρχείο ρυθμίσεων του συστήματος και του συγκεκριμένου λογαριασμού
- Οι τιμές των μεταβλητών ισχύουν
  - έως το τέλος του session, δηλ. μέχρι να τερματιστεί το bash shell
  - ή μέχρι να τις διαγράψει ο χειριστής
    - **unset <variable name>**

```
HOME    # The path to your home directory
TERM    # The terminal type
```

Μεταβλητές Περιβάλλοντος

- Μπορούμε να χρησιμοποιούμε τις μεταβλητές από τη γραμμή εντολών
  - Χρησιμοποιούμε τον τελεστή $

```
bash-3.00# myvar="hello"; echo $myvar
hello
bash-3.00# myvar="ls -la"
bash-3.00# $myvar
drwxr-xr-x2   ako2   staff     68    16 Jan 13:44 Applications
drwx------33  ako2   staff     1122  29 Mar 12:32 Desktop
drwx------21  ako2   staff     714   20 Mar 11:55 Documents
bash-3.00#
```

Ειδικές Μεταβλητές

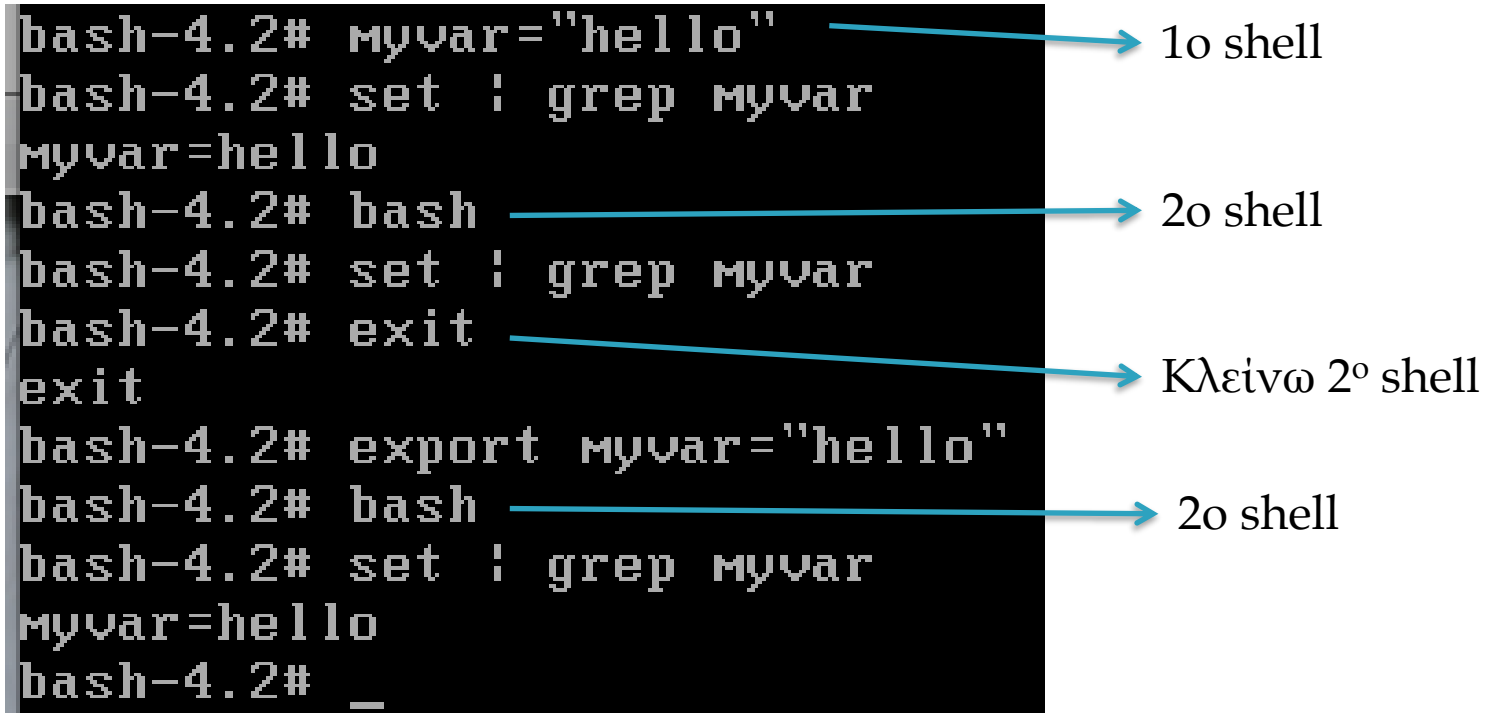
Εντολή	Περιγραφή
USER	Όνομα λογαριασμού χρήστη
HOME	Προσωπικός φάκελος χρήστη
TERM	Τύπος τερματικής συσκευής
SHELL	Ονομασία κελύφους
PATH	Λίστα φακέλων με εκτελέσιμες εντολές
MANPATH	Λίστα φακέλων με σελίδες βοήθειας (manual pages)
PWD	Τρέχων φάκελος
OLDPWD	Προηγούμενος τρέχων φάκελος
HOSTNAME	Ονομασία συστήματος

Χειρισμός Μεταβλητών

- Οι εντολές *env, printenv* εμφανίζουν την λίστα με τις ΚΑΘΟΛΙΚΕΣ μεταβλητές
- Η εντολή *set* εμφανίζει τη λίστα με τις ΤΟΠΙΚΕΣ και τις ΚΑΘΟΛΙΚΕΣ μεταβλητές
- Για να ορίσουμε μια ΚΑΘΟΛΙΚΗ μεταβλητή χρησιμοποιούμε την εντολή *export*
- Δήλωση μεταβλητών σύμφωνα με το περιεχόμενο
  - **String variables** -- myvar = "value”
  - **Integer variables** -- declare -i myvar
  - **Constant variables** -- readonly UnixRules=“YES"
  - **Array variables** -- declare -a MYARRAY MYARRAY[0]="one"; MYARRAY[1]=5; echo ${MYARRAY[*]}
- Τα ονόματα των μεταβλητών είναι case-sensitive
- Η εντολή *unset* διαγράφει μια μεταβλητή

Τοπικές – Καθολικές Μεταβλητές

- Για να ορίσουμε μια ΚΑΘΟΛΙΚΗ μεταβλητή χρησιμοποιούμε την εντολή *export*



Δημιουργία Νέων Εντολών

- Μπορούμε να δημιουργήσουμε νέες εντολές
  - Σε ένα αρχείο κειμένου εισάγουμε τις εντολές
- Για να τις εκτελέσουμε
  - Είτε με τη χρήση του *bash*
  - Ή κάνουμε το αρχείο εκτελέσιμο και το καλούμε απευθείας

```
bash-3.00# echo ’ls | wc -l’ > nu
bash-3.00# cat nu
ls | wc -l
bash-3.00# sh nu
     1
bash-3.00# bash nu
     1
bash-3.00# chmod a+x nu
bash-3.00# ./nu
     1
```

Χειρισμός Παραμέτρων [1/2]

- Μπορούμε να περάσουμε παραμέτρους σε ένα script
  - Ονομάζονται command-line arguments
- Χρησιμοποιούμε τις παραμέτρους σαν μεταβλητές

Παράμετρος	Περιγραφή
`$0`	Το όνομα του script / εκτελέσιμου
`$1` … `$9`	Η τιμή της 1ης … 9ης παραμέτρου
`$#`	Το πλήθος των παραμέτρων
`$*`	Όλες οι παράμετροι σαν string

```
bash-3.00# cat nu
echo Files found: `ls -la $1* | wc -l` "($1\*)"
bash-3.00# nu /b
Files found: 57 (/b*)
```

Χειρισμός Παραμέτρων [2/2]

- Για να χειριστούμε περισσότερες από 9 παραμέτρους
  - Δεν μπορούμε να χρησιμοποιήσουμε το $10
- Χρησιμοποιούμε την εντολή *shift x*
  - Μεταφέρει τις παραμέτρους προς τα αριστερά κατά *x* θέσεις
- Προσοχή – οι παλιές παράμετροι χάνονται

```
bash-3.00# cat ten
echo $* " -- " $#
shift 10
echo $* " -- " $#
bash-3.00# ten 1 2 3 4 5 6 7 8 9 10 11 12 13
1 2 3 4 5 6 7 8 9 10 11 12 13 -- 13
11 12 13 -- 3
```

Είσοδος από τον χειριστή

- Μπορούμε να ζητήσουμε είσοδο με τη χρήση της εντολής *read*
- Η σύνταξη είναι read var-name
- μπορούμε να ζητήσουμε πολλαπλές μεταβλητές
  read var1 var2 ...
- μπορούμε να εμφανίσουμε ένα μήνυμα πριν ζητήσουμε είσοδο
  read -p "Enter value:" var

```
bash-3.00# read -p "Enter values:" i j k;\
echo i=$i, j=$j, k=$k
abc d e f
i = abc, j = d, k = e f
```

Μαθηματικές Εκφράσεις

- Δυνατότητα μαθηματικών εκφράσεων με ακέραιους
  - Σχεδόν όπως στην C
  - Δεν χρειάζεται να έχουμε δηλώσει ότι η μεταβλητή είναι integer
  - Χρησιμοποιούμε την *expr* αντί για *atoi*

```
$ a=3
$ ((a = a + 1)) ; echo $a      //(4)
$ a=$((a+1))    ; echo $a      //(4)
$ a=$(($a+1)) ; echo $a        //(4)
$ a=a + 1; echo $a             //a+1
$ a=$a + 1; echo $a            //4+1

# ----το ίδιο με χρήση let----
$ let a = a + 1
$ let a++
$ a=`expr $a + 1`
```

Συνθήκη Ελέγχου – if και test

- Η εντολή ***test*** επιτρέπει την αποτίμηση μιας έκφρασης
  - Επιστρέφει **true** ή **false**
  - Προσφέρει μεγάλο εύρος εκφράσεων, π.χ., σχετικά με file permissions
    - if test -w "$1"; then echo "File $1 is writable“; fi

- Έχει δύο ισοδύναμους τρόπους σύνταξης:
  - **test** *expression*
  - **[** *expression* **]**

```
if [ condition 1 ]; then
  if [[ condition 2 && condition
3]]; then
  ...
  fi
elif [ condition 4 ] || [
condition 5 ] ; then
  ...
else ...
fi
```

Τελεστές test

Εντολή	Περιγραφή
`-gt`	greater than
`-ge`	greater than or equal
`-lt`	less than
`-le`	less than or equal
`-eq`	equal
`-ne`	not equal
`-n str`	non-empty string
`-z str`	zero-length string
`-d file`	το file είναι φάκελος (directory)
`-s file`	το file δεν έχει μηδενικό μέγεθος
`-f file`	το file υπάρχει
`-r file`	το file υπάρχει, και έχουμε read access
`-w file`	το file υπάρχει, και έχουμε write access
`-x file`	το file υπάρχει, και έχουμε execute access

Παράδειγμα if και test (1)

```
bash-3.00# cat check.sh
#!/bin/bash
read -p "Enter a filename: " filename
if [ ! -w "$filename" ]; then
  echo "File is not writeable"
  exit 1
elif [ ! -r "$filename"  ] ; then
  echo "File is not readable"
  exit 1
fi ...
```

Παράδειγμα if και test (2)

```
bash-3.00# cat check.sh
#!/bin/bash
TMPFILE = "diff.out"
diff $1 $2 > $TMPFILE
if [ ! -s "$TMPFILE" ]; then
  echo "Files are the same"
else
  more $TMPFILE
fi
if [ -f "$TMPFILE" ]; then
  rm -rf $TMPFILE
fi
```

Τελεστές boolean

```
if [ condition 1 && condition a ]; then
  if [ condition 2 || condition b ]; then
  ...
  fi
elif [ ! condition 3 ] ; then
  ...
else ...
fi
```

Συνθήκη Ελέγχου – case

```
case STRING in
pattern 1 )
   ... ;;
pattern 2 | pattern 3)
   ... ;;
*)
  echo "None of the above";;
   ...
esac
```

Παράδειγμα Ελέγχου – case

```
#!/bin/bash
read -p "Enter command: " command
case $command in
all | ALL )
  echo "Display all files..."
  ls -la;;
list | LIST)
  echo "Display all non-hidden files ..."
  ls -l;;
*)
  echo "Invalid choice"
  ls;;
esac
```

Βρόγχος – for

```
for VAR in <list>
do
... done
```

```
for i in 6 3 1 2
do
  echo $i
done | sort -n
```

```
for i in *.c do
echo $i done
```

Στους βρόγχους μπορούμε να χρησιμοποιήσουμε **break** και **continue** όπως κάνουμε στην C

Βρόγχος – while

```
while [ expression ];
do
...
done
```

```
i=1
while [[ $i -lt 10 ]]; do
  echo $i
  ((i++))
done
```

```
while true; do
  echo "alive..."
  sleep 3
done
```

Βρόγχος – until

```
until [ expression ];
do
...
done
```

```
Stop = "N"
until [[ $Stop = "Y" ]];
do
  ps -ef
  read -p "Do you want to stop?
(Y/N)" Stop
done
echo "Stopping..."
```

Παράδειγμα Script στο κέλυφος BASH

```
$ IFS=:  # Ορίζει το ":" ως διαχωριστικό tokens
$ for dir in $PATH
> do
>   if [ -x $dir/gcc ]   # Μην ξεχάσετε τα κενά!!
>   then
>       echo Found $dir/gcc
>       break
>   else
>       echo Searching $dir/gcc
>   fi
> done
```

- Για κάθε φάκελο που ορίζεται στη μεταβλητή περιβάλλοντος $PATH
  Έλεγξε αν περιέχει το εκτελέσιμο αρχείο gcc
  - Αν υπάρχει εκτύπωσε το *path* και σταμάτα
  - Αλλιώς συνέχισε την αναζήτηση στον επόμενο φάκελο

Συναρτήσεις

- Όλες οι συναρτήσεις πρέπει να οριστούν στην αρχή του script
- Μπορεί να μην έχουν παραμέτρους
- Οι παράμετροι και η τιμή που επιστρέφουν μπορεί να είναι από οποιονδήποτε τύπο
- Οι μεταβλητές που ορίζονται μέσα στη συνάρτηση είναι καθολικές!
  - Πρέπει να δηλώσουμε ότι είναι local

```
function name [()]
{
...
[return] }
```

Παράδειγμα Συναρτήσεων

```
#!/bin/bash
outside = "a global variable"
function mine() {
  local inside="this is local"
  echo $outside
  echo $inside
  outside = "a global with new value"
}
echo $outside
mine
echo $outside
echo $inside
```

Αποθήκευση και εκτέλεση

- Γράφουμε τις εντολές σε ένα αρχείο με οποιονδήποτε κειμενογράφο
  - Καλύτερα με τον **vi** (ή **vim**)
    - http://www.openvim.com/
    - https://blog.interlinked.org/tutorials/vim_tutorial.html
    - http://vim-adventures.com/
  - Εναλλακτικά με emacs, sublime, ή άλλον text editor

- Σώζουμε το αρχείο με extension **.sh**

- Πάμε στον φάκελο που περιέχει το αρχείο και εκτελούμε

```
$ bash myscript.sh
```

- Ή εναλλακτικά, αν έχουμε πρώτα δώσει eXecute privilege (**chmod +x myscript.sh**), εκτελούμε απευθείας

```
$ ./myscript.sh
```