Προγραμματισμός Bash κελύφους

- Πιο εύκολο από C κέλυφος
- Ακολουθία από εντολές
- # δηλώνει σχόλιο
- Πρώτη γραμμή #!/bin/bash
- Μεταβλητές, συνθήκες
- Δομές ελέγχου
- Ορίσματα προγράμματος
- Αριθμητικές πράξεις

Και άλλα πολλά…

Κατεβάστε το myscripts.tar από τη σελίδα του μαθήματος (εκεί όπου υπάρχουν και τα βιβλία)

Παράμετροι Προγράμματος – Πρόγραμμα parameters

```
#!/bin/bash
# all scripts start like this

#This is a comment

#will give 11 arguments to this program
# a b c d e f g h i j k

echo Number of input parameters = $#   # 11
echo Program Name = $0                 # ./parameters

echo Other Parameters = $1 $2 $3 $4 $5 $6 $7 $8 $9 $10 $11
#Other Parameters = a b c d e f g h i a0 a1

echo Other Parameters = $1 $2 $3 $4 $5 $6 $7 $8 $9 ${10} ${11}
#Other Parameters = a b c d e f g h i j k

echo All Arguments = $*
#All Arguments = a b c d e f g h i j k
```

Περί Μεταβλητών - Πρόγραμμα variables

```
#!/bin/bash

# POTE DEN BAZOUME TO '$' mprosta se mia metablhth?
# Apanthsh: Otan ths ana8etoume timh

#NEVER USE SPACES BEFORE AND AFTER = IN ASSIGNMENTS
a=2334              # Integer - Only digits
echo a              # a
echo $a             # 2334

hello="A B  C    D"
echo $hello         # A B C D
echo "$hello"       # A B  C   D
# Ta dipla eisagwgika diathroun ta polla kena

echo '$hello'       # $hello
# Ta mona eisagwgika apenergopoioun thn anafora timhs me $

echo -n "Enter \"b\" "       # Grafw lala koko
read b
echo "The value of \"b\" is now $b"
# The value of "b" is now lala koko

echo ${PATH}        # SWSTO - Metablhth periballontos PATH
```

Αριθμητικές Πράξεις - Πρόγραμμα arithmetics

```
#!/bin/bash

a=2334
let b=a+3  # isxyei kai to b=$a+3
let "c = a+3"
let "d = a+ 3"

z=$(($a+3))
y=$((a+3))  # Epishs swsto

k=`expr $a + 3` # Xrhsh entolhs expr

echo $a $b $c $d $k $z $y
#2334 2337 2337 2337 2337 2337 2337
```

Πολλές επιλογές. Για απλές πράξεις, οι εντολές let και expr. Για δεκαδικούς, η εντολή bc

Εντολή expr - Πρόγραμμα myexpr

```
#!/bin/bash

# PROSOXH: APAITOYNTAI KENA
a=`expr 3 + 5`        # 8
a=`expr 5 % 3`        # 2
a=`expr 5 / 3`        # 1
a=`expr 1 / 0` # Epistrefei sfalma
a=`expr 5 \* 3`       # 15
# Me to expr, ston pollaplasiasmo \*
a=`expr $a + 5` # Idio me let a=a+5

string=EnaMegaloString
position=4
length=6
z=`expr substr $string $position $length`
#E3agei length xarakthres apo to string.
#3ekinaei apo th 8esh position

echo $z  # Megalo
```

Πράξεις δεκαδικών με bc - Πρόγραμμα mybc

```
#!/bin/bash

# EPITREPEI ARI8MHTIKES PRA3EIS SE DEKADIKOUS
a=100.19
b=$(echo "scale=3; $a/100" | bc)
# scale ka8orizei dekadika pshfia

echo b = $b # b = 1.001

#perform inequality tests
A=0.04
B=0.03
let "comp=`echo $A-$B\>0 | bc`"
echo $comp       # 1

let "comp=`echo $B-$A\>0 | bc`"
echo $comp       # 0
```

Τιμή εξόδου – Πρόγραμμα exitStatus

```
#!/bin/bash

# To $? epistrefei ton kwdiko e3odou ths teleytaias
# entolhs pou ektelesthke
echo hello
echo $?      # 0 : epitygxhmenh ektelesh

lsdlsd       # agnwsth entolh
echo $?      # 127 - genikws mh mhdenikh se sfalma

echo

exit 113     # Prepei na einai 0-255
```

Συνθήκες

- Στη γενική μορφή σε δύο είδη:
  - [συνθήκη ελέγχου] ή
  - test συνθήκη ελέγχου

- Πιο σπάνια και αριθμητικές συνθήκες, συνθήκες [[ ]] …

Αριθμητικές Συνθήκες – Πρόγραμμα arithmeticTests

```
#!/bin/bash

# Arithmetic tests.
# The (( ... )) construct evaluates and tests
# numerical expressions.
# Exit status opposite from [ ... ] construct!

(( 0 ))
echo "Exit status of \"(( 0 ))\" is $?." # 1
(( 1 ))
echo "Exit status of \"(( 1 ))\" is $?." # 0
(( 5 > 4 )) # true
echo "Exit status of \"(( 5 > 4 ))\" is $?." # 0
(( 5 > 9 )) # false
echo "Exit status of \"(( 5 > 9 ))\" is $?." # 1
(( 5 - 5 )) # 0
echo "Exit status of \"(( 5  5 ))\" is $?." # 1
(( 5 / 4 )) # Division o.k.
echo "Exit status of \"(( 5 / 4 ))\" is $?." # 0
(( 1 / 2 )) # Division result < 1.
echo "Exit status of \"(( 1 / 2 ))\" is $?."
# Division is rounded off to 0.
# 1
(( 1 / 0 )) 2>/dev/null # Illegal division by 0.
#           ^^^^^^^^^^^^
echo "Exit status of \"(( 1 / 0 ))\" is $?." # 1
# What effect does the "2>/dev/null" have?
# What would happen if it were removed?
# Try removing it, then rerunning the script.
exit 0
```

Συνθήκες Αρχείων – Πρόγραμμα fileTests

```
#!/bin/bash

if [ -e fileTests ]    # exists file
  then if [ -f fileTests ] # is a regular file
         then echo Regular File
       fi
fi

# Omoia, to -d elegxei an prokeitai gia katalogo


if [ -r fileTests ]    # have read rights
   then echo I can read this file!!!
fi

# Omoia to -w kai -x
```

Συνθήκες Ακεραίων

-eq if [“$a” –eq “$b”]	Ίσα
-ne if [“$a” –ne “$b”]	Άνισα
-gt if [“$a” –gt “$b”]	Μεγαλύτερο ((“$a” > “$b”))
-ge if [“$a” –ge “$b”]	Μεγαλύτερο ή ίσο ((“$a” >= “$b”))
-lt if [“$a” –lt “$b”]	Μικρότερο ((“$a” < “$b”))
-le if [“$a” –le “$b”]	Μικρότερο ή ίσο ((“$a” <= “$b”))

Συνθήκες Strings (πάντα να χρησιμοποιείτε εισαγωγικά)

= if [“$a” = “$b”]	Ίσα
== if [“$a” == “$b”]	Ίσα
!= if [“$a” != “$b”]	Διαφορετικά
< if [“$a” \< “$b”]	Μικρότερο αλφαβητικά
> if [“$a” \> “$b”]	Μεγαλύτερο αλφαβητικά
-n if [-n “a”]	Όχι null
-z if [-z “a”]	Null (μέγεθος 0)

Λογικές Συνθήκες

! if [! –f “file”]	Λογικό NOT
-a if [“$a” –a “$b”]	Λογικό AND
-o if [“$a” –o “$b”]	Λογικό OR

Δομή Ελέγχου if

if <συνθήκη1>
then
        εντολές
elif <συνθήκη2>
then
        εντολές
…
else
        εντολές
fi

Προφανώς τα τμήματα elif και else είναι προαιρετικά

Δομή Ελέγχου case

case “$variable” in
“$condition1”)
εντολές….
;;
…
“$conditionN”)
εντολές….
;;
esac

Παράδειγμα Δομής case – Πρόγραμμα math

Πρόγραμμα που εκτελεί απλές πράξεις μεταξύ 2 ακεραίων.

```
#!/bin/bash
#
# Usage: math n1 op n2
#
case "$2" in
+) echo "Addition requested."
echo "$1 + $3 = `expr $1 + $3`" ;;
-) echo "Substraction requested."
echo "$1 - $3 = `expr $1 - $3`" ;;
\*) echo "Multiplication requested."
echo "$1 * $3 = `expr $1 \* $3`" ;;
/) echo "Division requested."
echo "$1 / $3 = `expr $1 / $3`" ;;
%) echo "Modulo arithmetic requested."
echo "$1 % $3 = `expr $1 % $3`" ;;
*) echo "Unknown operation specified." ;;
esac
```

Δομή for – Πρόγραμμα forLoops

```
#!/bin/bash

for koko in 1 2 3 4 5
do
  echo $koko
#Ektypwsh se diaforetikes grammes
done

for koko in "1 2 3 4 5"
do
  echo $koko
#Ektypwsh se mia grammh
done

NUMS="1 2 3 4 5"
for koko in $NUMS
do
  echo $koko
#Ektypwsh se diaforetikes grammes
done

for koko in `echo $NUMS`
do
  echo $koko
#Ektypwsh se diaforetikes grammes
done

LIMIT=5
#Diples parentheseis, LIMIT xwris $
for ((koko=1; koko <= 5; koko++))
do
  echo $koko
#Ektypwsh se diaforetikes grammes
done
```

Δομή for– Πρόγραμμα forLoops2

```
#!/bin/bash

#Xwris lista timwn epe3ergazetai
#tis parametrous tou programmatos
for koko
do
  echo -n $koko
done
echo

#how to parse some arguments
#from $2 until the end
for j in ${*:2}
do
  echo -n $j
done
echo

#$2 to $4
#start at position 2 and use
#3 arguments
for j in ${*:2:3}
do
  echo -n $j
done
echo
```

Παράδειγμα– Πρόγραμμα breakCont

```
#!/bin/bash
LIMIT=19 # Upper limit
echo
echo "Numbers 1 through 20 (but not 3 and 11)."
a=0
while [ $a -le "$LIMIT" ]
do
a=$(($a+1))
#Agnohse ta 3, 11
if [ "$a" -eq 3 ] || [ "$a" -eq 11 ]
then
continue
fi
echo -n "$a " # Den ekteleitai gia ta 3 and 11.

done

echo

a=0
while [ "$a" -le "$LIMIT" ]
do
a=$(($a+1))
if [ "$a" -gt 2 ]
then
break # Skip entire rest of loop.
fi
echo -n "$a "
done

echo
```

Δομή while – Πρόγραμμα whileLoops

```
#!/bin/bash
var0=0
LIMIT=10
while [ "$var0" -lt "$LIMIT" ]
do

echo -n "$var0 "
var0=`expr $var0 + 1`
# var0=$(($var0+1)) also works.
# var0=$((var0 + 1)) also works.
# let "var0 += 1" also works.

done
echo
exit 0
```

Η εντολή “set -- $var” Πρόγραμμα setProg

```
#!/bin/bash

echo Input parameters = $#
var="one two three four five"

#split based on blank chars
#assign to input parameters!!
set -- $var

echo Input parameters = $#
#Now prints 5

for koko
do
  echo $koko
done
```

Παράδειγμα – Πρόγραμμα revstrs

Πρόγραμμα που τυπώνει αντίστροφα τις συμβολοσειρές εισόδου του, καθώς και το μήκος τους

```
#!/bin/bash
#
# Usage: revstrs [string1 [string2 ...]]
#
for str
do
strlen=`expr length "$str"`
# &a arxhsoume ektypwsh apo to telos
# Prepei na 3eroume mhkos

chind=$strlen
while test $chind -gt 0
do
echo -n "`expr substr \"$str\" $chind 1`"
chind=`expr $chind - 1`
done
echo -n " --> "
echo -n "$strlen"
echo " character(s)."
done
```

Παράδειγμα – Πρόγραμμα listRegFiles

Ονόματα κανονικών αρχείων εντός ενός καταλόγου

```
#!/bin/bash

OUTFILE=files.lst

dirName=${1-`pwd`}
echo DIRNAME = $dirName

# To - dhlwnei default timh
# An den dw8ei onoma katalogou
# apo xrhsth

echo Regular files in directory\
      "$directory" > $OUTFILE

# -type f means regular files
for file in "$( find $dirName -type f )"
do
  echo "$file"
done | sort >> "$OUTFILE"
#      ^^^^^^^^^^^^^^^^^^^^
# Anakateu8ynsh ta3inomhmenou stdout
```

Παράδειγμα – Πρόγραμμα shiftCommand

Επεξεργασία ορισμάτων προγράμματος 1-1. Εντολή shift

```
#!/bin/bash

# call with > 5 arguments

echo Args1 = $*

for str  # prints OK even with change
do
var=$1
shift
echo "var = $var and args = $*"

done
```

Παράδειγμα – Πρόγραμμα factorial

Υπολογίστε το παραγοντικό του αριθμού που περνιέται σαν όρισμα στο πρόγραμμα

```
#!/bin/bash
#
# Usage: factorial number
#
if [ "$#" -ne 1 ]
then
echo "Just give 1 argument"
exit 1
fi

if [ "$1" -lt 0 ]
then
echo Please give positive number
exit 1
fi

fact=1
for ((i = 1; i <= $1; i++))
do
fact=`expr $fact \* $i`
done
echo $fact
```

Παράδειγμα – Πρόγραμμα dirSize

Μέγιστο μέγεθος καταλόγου από αυτούς που δίνονται ως ορίσματα

```
#!/bin/bash
#
# Usage: dirSize dirName1 ... dirNameN
#
max=0
maxdir=$1
for dir
do
if [ ! -d $dir ]
then
echo "No directory with name $dir"
else
size=`du -sk $dir | cut -f1`
echo "Size of dir $dir is $size"
if [ $size -ge $max ]
then
max=$size ; maxdir=$dir
fi # if size...
fi # if directory
done
echo "$maxdir $max"
```

Παράδειγμα – Πρόγραμμα printContents

```
#!/bin/bash
# Loads this script into an array.

text=( $(cat "$0") )
for element in $(seq 0 $((${#text[@]} - 1)))
do
# ${#text[@]}
#+ gives number of elements in the array.
echo -n "${text[$element]}"
# Each field of this script on a single line.
echo -n " .. " # Seperate by " .. "
done

echo

for ((i=0; i <= ${#text[@]} - 1; i++))
do
echo -n "${text[$i]}"
# Each field of this script on a single line.
echo -n " .. " # Seperate by " .. "
done

echo

for i in `cat "${0}"`
do
echo -n "${i}"
# Each field of this script on a single line.
echo -n " .. " # Seperate by " .. "
done

echo
```

Παράδειγμα – Πρόγραμμα printContents2

Διάβασμα αρχείου γραμμή-γραμμή

```
#!/bin/bash
exec < "$0" #Take input from this file
while read line
do
echo $line
done

#IFS is an internal variable specifying
#how bash separates fields, word boundaries
#ALWAYS SAVE TO TEMP VARIABLE AND
#RESET AFTERWARDS
OLDIFS="$IFS"
IFS=$'\n'       #IFS=       also works
for line in `cat "$0"`
do
echo "$line"
done
IFS="$OLDIFS"

exit 0
```

Παράδειγμα – Πρόγραμμα listAndCopy

Εύρεση *.h αρχείων σε κατάλογο. Αποθήκευση 3 πρώτων γραμμών τους σε αρχείο myout

```
#!/bin/sh
#search for .h files in a
#directory
#For each file it lists
#the first 3 lines in the file
#myout

FILE_LIST=`ls /usr/include/c++/4.0.3/ext/*.h`
touch myout
rm myout
touch myout

for file in ${FILE_LIST}
do
echo FILE = ${file}
    head -3 "${file}" >> myout
done
```

Παράδειγμα – Πρόγραμμα countword

Διάβασμα αρχείου. Αφαίρεση πολλαπλών συνεχόμενων λέξεων. Εκτύπωση κάθε λέξης στη μορφή λέξη/#συνεχόμενες_εμφανίσεις

```
#!/bin/bash
# Loads this script into an array.
prev=""
cons=1

for str in `cat ${1}`
do

if [ "${str}" != "$prev" ]
then
if [ ! -z $prev ]
then
echo "${prev}/${cons} "
fi

prev=${str}
cons=1
else
let "cons = cons + 1"
fi
done

if [ ! -z prev ]
then
echo "${prev}/${cons}"
fi
```