



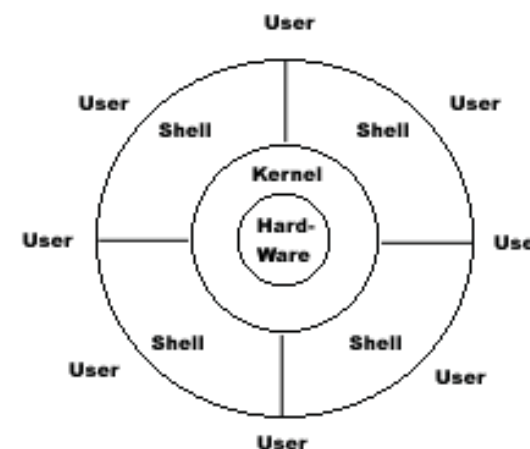
Διάλεξη 8 Το Κέλυφος Bash

Δημήτρης Ζεϊναλιπούρ



Περιεχόμενο Διάλεξης

- Εισαγωγή στο Κέλυφος Bash
- Περιβάλλον
 - Περιβάλλον και Μεταβλητές
 - Αρχεία ελέγχου Login/Logout
 - Μεταβλητές Κελύφους
 - Εντολές Κελύφους
- Προγραμματισμός Κελύφους
 - Εκτέλεση Προγραμμάτων
 - Είσοδος Τιμών από τον Χρήστη
 - Command-line Parameters



Εισαγωγή στο Κέλυφος Bash



- Πήρε το όνομα από το **Bourne Again SHell**
- Δημιουργήθηκε για να παρέχει μια **ανοικτή υλοποίηση (open source)** του Bourne Shell.
- Παρέχει καλύτερη **διαχείριση εργασιών, συναρτήσεις, aliases**, κτλ., και μπορεί να **εκτελεί Bourne scripts** χωρίς αλλαγές
- **Λειτουργίες Bash:**
 - **Διερμηνέας Εντολών** (command interpreter) / **Εκτελέσιμο Αρχείο** (μπορεί να εκτελεστεί σαν μια οποιαδήποτε άλλη εντολή **\$/bin/bash**)
 - **Γλώσσα Προγραμματισμού** (script programming language)

Εισαγωγή στο Κέλυφος Bash



Το Bash χρησιμοποιείται κατά τους ακόλουθους τρεις τρόπους :

- **Διαδραστικά (Interactive Execution):** δίδοντας εντολές στο command line
- **Μαζική Εκτέλεση (Batch Execution):**
Τοποθετήστε μια ακολουθία εντολών σε ένα αρχείο **run.sh**, το οποίο περιέχει το **hash-bang** **#!/bin/bash** στη πρώτη γραμμή και εκτελέστε (chmod +x run.sh; ./run.sh)
- **Δομημένος Προγραμματισμός (Structured Programming):** Με την χρήση εντολών επιλογής (if), Δομών επανάληψης, Συναρτήσεων κτλ., που βρίσκουμε σε γλώσσες υψηλού επιπέδου)

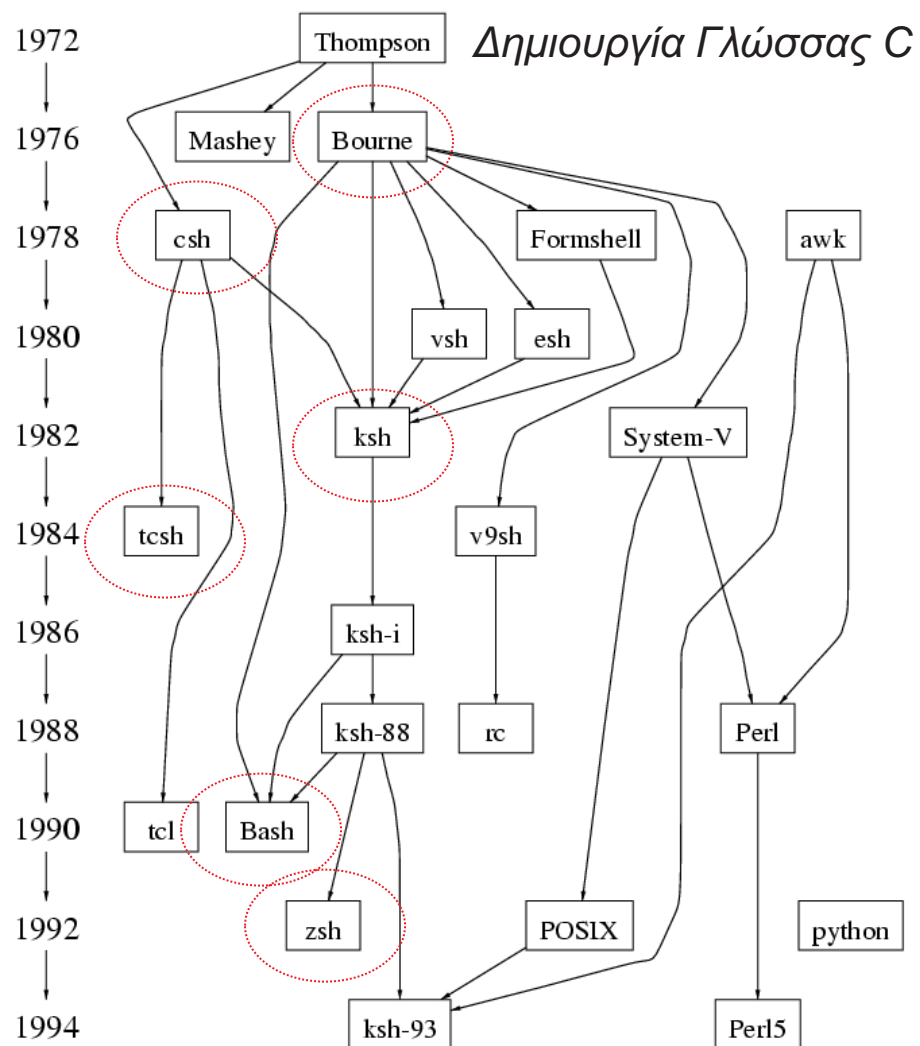


Η Εξέλιξη των Κελυφών

- **1976** Ο Steven Bourne αναπτύσσει το **Bourne Shell (sh)** στα εργαστήρια της Bell Labs. (1973: Το UNIX γράφεται σε γλώσσα C)
- **1978** Ο Bill Joy αναπτύσσει το **C shell (csh)** στο University of California – Berkeley.
- **1982** Ο David Korn αναπτύσσει το **Korn Shell (ksh)** στα εργαστήρια της Bell Labs.
- **1984** Ο Ken Green (Carnegie Mellon) και Paul Placeway (Ohio State), αναπτύσσουν το **TENEX C shell (tcsh)**, ως επέκταση του **csh**, προσφέροντας την δυνατότητα για filename completion και άλλα.
- **1990** Ο Brian Fox αναπτύσσει το **Bourne-Again-Shell (bash)**, ένα υπερσύνολο του **sh**, το οποίο σήμερα είναι το πιο διαδεδομένο κέλυφος σε Linux και MacOSX και είναι υλοποιημένο για τις πλείστες πλατφόρμες Unix καθώς και Windows/cygwin
- **1992** Ο Paul Falstead αναπτύσσει το **zsh**, το οποίο είναι επέκταση του **sh** με διάφορες επιπλέον δυνατότητες που βρίσκουμε στο **ksh**, **csh** και **bash**



Η Εξέλιξη των Κελυφών



Ken Thompson

Dennis Ritchie



Το Κέλυφος BASH

- Εάν δεν είστε σίγουροι σε πιο κέλυφος βρίσκεστε κάποια δεδομένη στιγμή, μπορείτε να εκτελέσετε την εντολή:

```
$echo $SHELL / echo $BASH_VERSION  
– /bin/bash 3.1.17(1)-release
```

- Για να χρησιμοποιήσετε το bash shell πληκτρολογήστε :

```
$ /bin/bash  
bash-3.1$
```

- Μπορείτε εάν θέλετε να τροποποιήσετε το περιεχόμενο του command prompt, αλλάζοντας την μεταβλητή PS1

```
bash-3.1$ PS1="`pwd` $"  
/home/faculty/dzeina $
```

- (μπορεί να χρησιμοποιηθεί σε συνδυασμό με την export για να συνεχίσει το bash να είναι το κέλυφος σας σε ότι νέα κελύφη ανοίξετε κατά την διάρκεια μιας σύνδεσης)



Ορίζοντας το BASH ως το default shell

- Εάν το bash δεν είναι το default shell σας και θέλετε να το αλλάξετε (αυτή η διαδικασία ΔΕΝ ισχύει για το Π.Κ. (όπου αυτή η διαδικασία πρέπει να γίνει μέσω του LDAP) τότε:

- \$passwd -s (ή \$chsh)**

```
dzeina@abc:~>passwd -s
```

```
Current available shells:
```

```
/bin/sh
```

```
/bin/bsh
```

```
/bin/csh
```

```
.....
```

```
/usr/bin/tsh
```

```
/usr/bin/tcsh
```

```
/bin/bash
```

```
/usr/bin/bash
```

```
dzeina's current login shell:
```

```
/usr/bin/tcsh
```

```
Change (yes) or (no)? > yes
```

```
To?>/bin/bash
```

```
dzeina@abc:~>
```

Αυτό μας επιτρέπει να αλλάξουμε το default shell μέσα στο **/etc/passwd** αρχείο, το οποίο διατηρεί διάφορα στοιχεία (**username, passwd, userid, groupid, name, home directory και shell**). Τα κωδικοποιημένα συνθηματικά πρόσβασης των χρηστών του συστήματος είναι στο **/etc/shadow** (no read permissions)

```
bash-2.05$ cat /etc/passwd | grep dzeina
```

Μας δίνει:

```
dzeina:!:1240:243:Dimitris_Zeinalipour:/home/faculty/dzeina:/bin/bash
```

Προσοχή: Αν αλλάξετε το **default shell** και δεν έχετε τα αρχεία **.bash_profile** (επόμενες διαφάνειες) τότε το περιβάλλον του κελύφους δεν θα είναι ορθά ρυθμισμένο.

Εναλλακτικά απλά πληκτρολογείτε \$bash μετά από κάθε login

Περιβάλλον και Μεταβλητές



- Όταν κάνουμε login, εκτελούνται με την ακόλουθη σειρά τα scripts

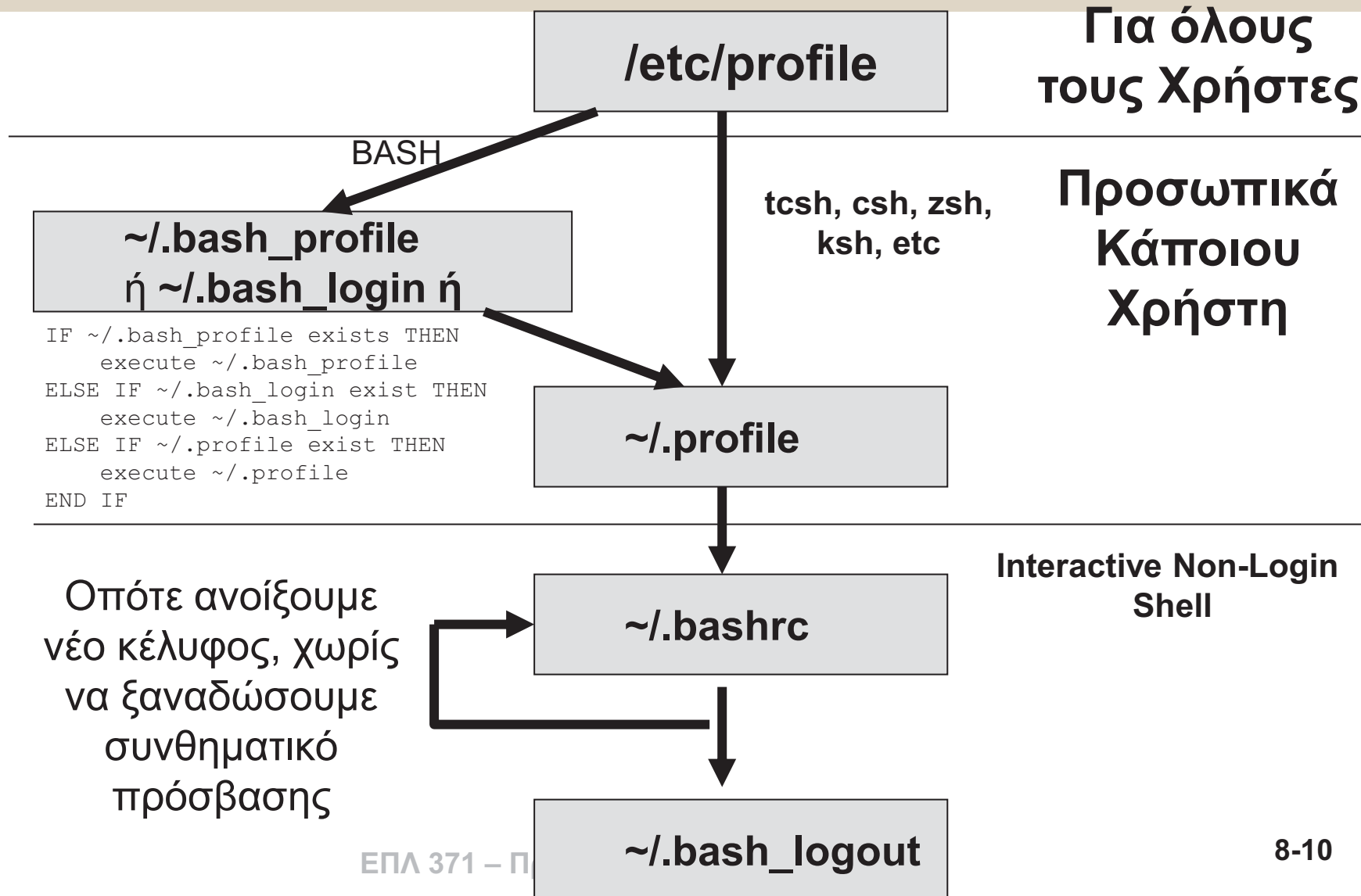
A) **/etc/profile**

B) **~/.bash_profile**, **~/.bash_login** ή **~/.profile**

C) **~/.bash_logout**



Αρχεία ελέγχου Login/Logout



Περιβάλλον και Μεταβλητές



A) Το αρχείο **/etc/profile**

- Αυτό είναι το system-wide startup file (για όλους τους χρήστες)
- Το κέλυφος διαβάζει και εκτελεί τις εντολές οι οποίες βρίσκονται στο αρχείο **/etc/profile**, για οποιονδήποτε χρήστη κάνει login.
- Για να μπορέσει κανείς να αλλάξει αυτό το αρχείο χρειάζεται να έχει δικαιώματα root.

Αρχεία ελέγχου Login/Logout

B) **\$HOME/.bash_profile** (Σε κάθε login)

- Ένα από τα πιο σημαντικά αρχεία του bash
- Εκτελείται ακριβώς μετά το **login**, εάν το default κέλυφος σας είναι το bash shell
- Περιέχει **βασικές μεταβλητές** και **ρυθμίσεις** του περιβάλλοντος του λογαριασμού σας, π.χ. :
 - PATH=/usr/local/bin:/usr/bin:/usr/ucb:\$HOME/bin
 - MANPATH=/usr/local/man:/usr/man
 - EDITOR=/usr/bin/vi
 - export PATH MANPATH EDITOR
 - MAIL=/var/mail/\$LOGNAME
- Εάν τροποποιήσετε το πιο πάνω αρχείο τότε για να **ενεργοποιηθούν** οι αλλαγές πρέπει να αποσυνδεθείτε (logout)/συνδεθείτε ξανά (login), ή να πληκτρολογήσετε:
. \$HOME/.bash_profile ή **source \$HOME/.bash_profile**
- Εκτέλεση εντολής μέσα στο ίδιο περιβάλλον του κελύφους.



Αρχεία ελέγχου Login/Logout

\$HOME/.bash_profile (continued)

- Διαβάζεται και εκτελείται μόνο κατά τη διάρκεια του login, και όχι εάν ανοίξετε ένα νέο κέλυφος.
- Εάν το **\$HOME/.bash_profile** δεν υπάρχει, τότε το bash θα ψάξει για το **\$HOME/.bash_login**. Εάν δεν υπάρχει ούτε αυτό το αρχείο τότε, θα ψάξει για το **\$HOME/.profile**
- Επομένως, μπορείτε να κρατάτε κάποιες κοινές ρυθμίσεις για όλα τα κελύφη στο **\$HOME/.profile** (κοινό για tcsh, sh, bash, etc)

Παράδειγμα `$HOME/.bash_profile`



```
$cat ~/.bash_profile
```

```
# $HOME/.bash_profile
```

```
# Get the aliases and functions
```

```
if [ -f ~/.bashrc ]; then
```

```
    . ~/.bashrc
```

```
fi
```

Εάν υπάρχει αρχείο με το όνομα `~/.bashrc`, τότε εκτέλεσε την εντολή `~/.bashrc` (δηλαδή εφάρμοσε την αρχικοποίηση που εφαρμόζουμε και στο subshell – δες επόμενη διαφάνεια)

```
# User specific environment and startup programs
```

```
PATH=$PATH:$HOME/bin
```

```
BASH_ENV=$HOME/.bashrc
```

```
export BASH_ENV PATH
```

Αρχεία ελέγχου Login/Logout



B3) **\$HOME/.bashrc** (για subshells)

- Εκτελείται οπότε δημιουργήσετε ένα **bash subshell**, πληκτρολογώντας στο command line \$bash (ή εάν εκτελέσετε ένα bash shell program το οποίο είναι αποθηκευμένο σε αρχείο)
- Επομένως, μπορείτε να τοποθετήσετε όλες τις εντολές που χρειάζεστε κατά την διάρκεια του login στο **\$HOME/.bash_profile**, και ότι εντολές χρειάζεστε με την εκτέλεση ενός **subshell**, στο **\$HOME/.bashrc**

Αρχεία ελέγχου Login/Logout



C) `$HOME/.bash_logout` (για το `logout`)

- (Μπορεί) να βρίσκεται στο home directory του χρήστη.
- Εάν δεν υπάρχει, μπορείτε να το προσθέσετε μόνοι σας.
- Σημειώστε ότι αυτό το αρχείο εκτελείται οποτεδήποτε κάποιος εκτελέσει την εντολή `$logout`.
- Εάν κάποιος χρήστης απλά κλείσει το παράθυρο του κελύφους (ή το ssh connection), τότε **δεν εκτελείται**.



Αρχείο `.bash_history`

- Αυτό το αρχείο περιέχει τις εντολές τις οποίες έχετε εκτελέσει στο παρελθόν
- Το αρχείο ενημερώνεται όταν εξέλθετε του κελύφους με την **logout** ή όταν κλείσει η σύνδεση με το τερματικό.

`ls -a` → every process, full information

`ps -ef | awk '{print $1}' | sort | uniq`
`users`

.....

Μεταβλητές Κελύφους



- Οι μεταβλητές κελύφους μας επιτρέπουν να αποθηκεύουμε διαφορές τιμές.
- Οι **καθολικές μεταβλητές** δηλώνονται συνήθως με κεφαλαίους χαρακτήρες.
- Για να έχετε πρόσβαση στην τιμή μιας μεταβλητής χρησιμοποιήστε τον **τελεστή \$**
- Για να **εκτυπώσετε** το **περιεχόμενο** μια μεταβλητής, χρησιμοποιήστε την εντολή **echo**
- Παράδειγμα: **echo \$RANDOM**
 - **Αποτέλεσμα:** 29405 # εκτύπωση ψευδοτυχαίου αριθμού

Μεταβλητές Κελύφους



- Υπάρχουν δυο ειδών μεταβλητές **Καθολικές (Global)**, διαθέσιμες σε όλα τα υπό εκτέλεση κελύφη, και **Τοπικές (Local)**, διαθέσιμες στο κέλυφος που ορίζει την μεταβλητή.
- Εκτύπωση Μεταβλητών
 - Καθολικές : **env** ή **printenv**
 - Τοπικές : **set**
- Για να κάνετε γνωστή μια μεταβλητή σε **ΌΛΑ** τα sub-processes (υπό-κελύφη και προγράμματα που δημιουργούνται από ένα κέλυφος) χρησιμοποιήστε την εντολή :
export (π.χ. export MYVAR="something")

Σημαντικές Καθολικές Μεταβλητές



Μεταβλητή Shell	Περιγραφή
PWD	The most recent current working directory (e.g., /home/faculty/dzeina/public_html)
OLDPWD	The previous working directory (e.g., /home/faculty/dzeina/)
BASH	The full path name used of the bash shell. (e.g., /bin/bash)
RANDOM	Generates a random integer between 0 and 32,767.
HOSTNAME	The current hostname of the system (e.g., aias)
PATH	A list of directories to search of commands (A command needs to be in your PATH in order to execute it from any directory. Otherwise you need to execute it as ./command.)
HOME	The home directory of the current user. (cd \$HOME==cd ~)
PS1	The primary prompt. (e.g., dzeina@www2>)
PS2	Second level prompt. (e.g., \$)

Μεταβλητές Κελύφους (Χρήστη)



- Ένας άλλος τρόπος κατηγοριοποίησης των εντολών κελύφους είναι βάση του τύπου:
- **Εντολή (παράδειγμα τρόπος δήλωσης)**
 - String Variables (**myvar="info.txt"**)
 - Integer Variables (**declare -i myvar**)
 - Constant Variables (**readonly fn="info.txt"**)
 - Array Variables (**declare -a ARRAY** : θα τα δούμε αργότερα)



Μεταβλητές Κελύφους (Χρήστη)

- Σύνταξη: **variable_name = value**
- Παράδειγμα: Δημιουργήστε μια μεταβλητή με το όνομα “rate” η οποία αποθηκεύει το επιτόκιο και αρχικοποιήστε την σε 7.65.
\$ rate=7.65 Προσοχή: Case-sensitive Variables (διάκριση πεζών-κεφαλαίων)
- Η μεταβλητή δεν έχει τύπο (μέχρι στιγμής)
- Εάν η μεταβλητή έχει white spaces τότε χρησιμοποιούμε διπλά εισαγωγικά
π.χ. name=“Thomas William Flowers”



Διαγραφή Μεταβλητών

- Η εντολή **unset** μπορεί να χρησιμοποιηθεί για να διαγράψουμε **τοπικές** αλλά και **καθολικές** μεταβλητές περιβάλλοντος (προσωρινά)
- Η unset δουλεύει **για όλους τους τύπους**, εκτός το **readonly**, οι οποίες μεταβλητές δεν μπορούν να **διαγραφούν** μέχρι να κάνουμε **logout**.
- **Σύνταξη: unset variable_name**
- Παράδειγμα: Διαγραφή του MYPATH the contents of the variable MYPATH.

```
$ unset MYPATH
```

```
$ echo $MYPATH
```

```
$
```



Εξαγωγή (Export) Μεταβλητών

- Όταν δημιουργήσετε μια μεταβλητή, τότε αυτή η μεταβλητή δεν είναι γνωστή στα υπό-κελύφη ή υπό-προγράμματα
- Εάν θέλετε να κάνετε το περιεχόμενο μιας μεταβλητής γνωστό στα υπό-προγράμματα και υπό-κελύφη τότε χρησιμοποιήστε την εντολή `export`.
- Σύνταξη: **`export varname(s)`**
- Εάν θέλετε να δείτε όλες τις exported μεταβλητές τότε απλά πληκτρολογήστε **`export`**

Παράδειγμα Χρήσης Export



```
$ myvar="test"
```

```
$ set | grep myvar    # Εκτύπωση τοπικών μεταβλητών
```

```
myvar=test           # Το myvar περιέχει την τιμή test (τοπικά)
```

```
$ bash                # Δημιουργία sub-shell 2
```

```
$ set | grep myvar    # Εκτύπωση τοπικών μεταβλητών
```

```
$                    # Τίποτα δεν εκτυπώνεται, δεν υπάρχει το myvar
```

```
$ exit                # Επιστροφή στο Shell1
```

```
$ export myvar="test" # Αυτή την φορά την κάνουμε καθολική
```

```
$ set | grep myvar    # Εκτύπωση τοπικών μεταβλητών
```

```
myvar=test
```

```
$ bash                # Δημιουργία sub-shell 2
```

```
$ set | grep myvar
```

```
myvar=test           # Αυτή την φορά υπάρχει το myvar!
```



(Built-in) Εντολές Κελύφους

- Το κάθε κέλυφος έχει ένα σύνολο εντολών τα οποία είναι μέρος του κελύφους (***built-in commands***)
- Αυτές οι εντολές είναι compiled μέσα στο κέλυφος; Επομένως δεν θα τις βρείτε σε κάποιο directory (π.χ., /user/bin/)
- Όταν εκτελούνται αυτές οι εντολές εκτελούνται μέσα στο περιβάλλον της διεργασίας του bash, π.χ., το **\$history** δεν δημιουργεί νέα διεργασία
- Εάν θέλετε να μάθετε περισσότερα για αυτές τις (πολλές εκ' των οποίων ήδη γνωστές) εντολές πληκτρολογήστε **\$ bash -c help.**



Εκτέλεση Προγραμμάτων Shell

- Εάν εκτελέσουμε ένα shell script **script-name.sh**, τότε το κέλυφος δημιουργεί (fork) μια νέα διεργασία.
- Ενόσω το **script-name.sh** script εκτελείται, το κέλυφος περιμένει την εκτέλεση.
- Αυτό ισχύει στις ακόλουθες περιπτώσεις:
\$ **./script-name** ή \$**bash** script-name
(ή απλά script-name, εάν έχετε προσθέσει το μονοπάτι του script στην μεταβλητή περιβάλλοντος PATH)
- Εάν θέλετε να εκτελέσετε **scriptname.sh** μέσα στο περιβάλλον της διεργασίας του κελύφους, τότε μπορείτε να χρησιμοποιήσετε την εντολή **source**.
\$**source** script-name == \$ **.** script-name
Σημείωση: δεν συνιστάται στις πλείστες περιπτώσεις.



Εκτέλεση Προγραμμάτων Shell

1)

Shell:

source run.sh

date

Built-in

du

Built-in

2)

Shell:

run.sh

Subshell:

date

Built-in

du

Built-in

3)

Shell:

run.sh &

Εδώ μπορεί να εκτελεστούν άλλες
εντολές εφόσον το script είναι στο
background

Subshell:

date

Built-in

du

2,3: Προτιμότερο γιατί ότι τοπικές μεταβλητές δημιουργούνται στο subshell σβήνονται με την έξοδο από το script



Εντολές (Built-in) Κελύφους

Όλες οι built-in εντολές είναι διαθέσιμες με το **\$bash -c help**

Built-in command	Description	Example
. filename or source filename	Executes commands from specified filename in the current shell environment	. script
cd	Changes the current directory	cd /
declare or typeset	Declares a variable; the -r option makes the variable read-only	declare var2 <i>declare -r var3 ή readonly var3</i>
echo	Displays output on standard output	echo "Hello"
exec	Causes a command to replace the current shell; no new process is created	exec ls // Το ls παίρνει την θέση του bash (άρα χάνετε το κέλυφος)
exit	Causes the shell to exit with a status	exit 1



Εντολές (Built-in) Κελύφους

Built-in command	Description	Example
export	Exports the variable to a subshell	export rate="2.75" export TERM
history	Displays history of commands previously executed	history
kill	Sends a signal to a PID. To see all signals, run kill -l	kill 1409 kill -l
let	Evaluates an arithmetic expression	let x=5+10
local	Creates a local variable (useful for functions)	local x=5
logout	Exits a login shell	logout
pwd	Display current working dir	pwd
read	Reads characters from stdin	read reply

Εντολές (Built-in) Κελύφους



Built-in command	Description	Example
readonly	Once a variable is declared as readonly, it cannot be changed until logout	readonly y
return	Causes a function to exit with a value	return 1
set	Sets and displays shell variables (χρησιμοποιείτε στο debugging)	set -xv , set +xv
shift	Shifts positional parameters Εάν έχουμε περισσότερα από 9 command-line arguments. (θα δούμε παράδειγμα αργότερα)	shift 2
test	Evaluates expressions	test -d temp Same with [-d demo]
trap	Catches a signal sent to the shell	trap "echo Trapped" 3



Είσοδος Από τον Χρήστη - read

- Η εντολή **read** μας επιτρέπει να πάρουμε τιμές από τον χρήστη και να τις αποθηκεύσουμε σε μεταβλητές

- Σύνταξη:

read *var-name [more vars]*

ή

read -p "Please Enter:" *var-name [more vars]*

- Η τελευταία μεταβλητή λαμβάνει ότι περισσεύει!

Παράδειγμα

```
$read a b c ; echo a=$a,b=$b,c=$c
```

```
1 2 3 4 5 6
```

```
a=1,b=2,c=3 4 5 6
```



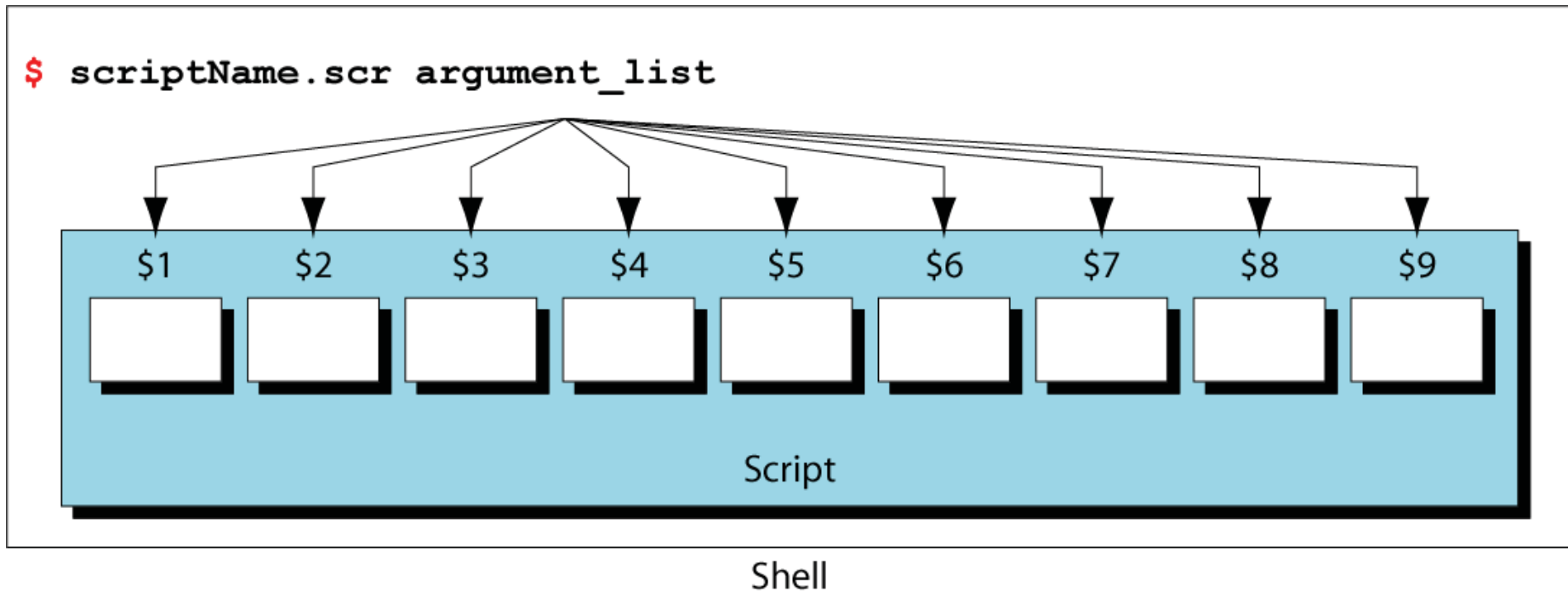


Command-line Arguments

Μπορείτε να χρησιμοποιήσετε command-line arguments για να περάσετε παραμέτρους σε ένα shell script.

Parameter	Meaning
\$0	References the name of the current shell script or a UNIX command. (το όνομα του shell script)
\$1-\$9	Command Line Argument 1 to 9
\$#	Ο αριθμός των command line arguments π.χ. <code>./run.sh "costas"</code> , επιστρέφει 1
\$*	Επιστρέφει τις παραμέτρους σαν μια συμβολοσειρά \$1 \$2 ...\$9.

Command Line Arguments





Παράδειγμα: Command Line Arguments

\$ bash -xv run.sh tim bill ann fred

\$1 \$2 \$3 \$4

echo \$*

+ echo tim bill ann fred
tim bill ann fred

echo \$#

+ echo 4
4

echo \$1

+ echo tim
tim

echo \$3 \$4

+ echo ann fred
ann fred

Debug

Show variable names

συνέχεια

Με κόκκινο είναι οι εντολές στο run.sh

ός Συστημάτων

8-35

Παράδειγμα: Command Line Arguments



Εκτέλεση χωρίς τον bash debugger

\$ run.sh tim bill ann fred

tim bill ann fred

4

tim

ann fred



Ειδικοί Χαρακτήρες και Εισαγωγικά

- **date=20072201**
- **echo \$date** : Εκτυπώνει 20072201
echo \ \$date: Εκτυπώνει \$date\
- **Single-Quote**: Διατηρεί την τιμή του κάθε χαρακτήρα που εσωκλείεται στις παρενθέσεις
echo '\$date': Εκτυπώνει \$date
- **Double Quote**: Διατηρεί την τιμή του κάθε χαρακτήρα εκτός και αν υπάρχει \$, ` (backward single quote) ή \
echo "\$date": Εκτυπώνει 20072201
- **Command substitution**: Εκτελείται η εντολή και το αποτέλεσμα εκτυπώνεται
echo `date` (**backward single quote**) : Εκτυπώνει Mon Jan 22 22:22:59 RST 2007
- **Posix-Compatible Command Substitution**: **echo \$(date)** ή
Ανάθεση Μεταβλητής: **i=\$(cat file.txt | grep string)**
- **echo sp{el,il,al}l**: Εκτυπώνει spell spill spall!
- **echo "Shell=\${SHELL?Undefined}"** Τύπωσε το περιεχόμενο της μεταβλητής SHELL ή εκτύπωσε την συμβολοσειρά Undefined



Ειδικοί Χαρακτήρες και Εισαγωγικά

- **echo "This will print **
as one line"
- **Escape Characters (printf, echo and sed)**
\n = newline
\t = tab
\b = backspace

Σημείωση:

Υποδηλώνει ότι θέλουμε να
εκτυπώσουμε τα escape characters

echo **-e** "A\tB" => Τυπώνει: A B

printf "%d\t%d" 20 30 => Τυπώνει: 20 30

Εκτύπωση στο Κέλυφος - printf



- Για να εκτυπώσουμε στο κέλυφος, υπάρχει, πέρα από την echo, επίσης η εντολή **printf**, ή οποία έχει όμοια λειτουργία με αυτή της C.

printf "format-string" [parameters]

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

```
PI=3.14159265358979; Age=20; Message1="Greetings,"  
Message2="Testing."
```

```
printf "Pi to 2 decimal places = %1.2f \n" $PI
```

```
printf "Age = \t%d\n" $Age
```

```
printf "%s %s \n" $Message1 $Message2
```

```
# sprintf() με POSIX-compatible command substitution
```

```
Pi2=$(printf "%1.2f" $PI); ή `printf "%1.2f" $PI`
```

```
echo "Pi to 2 decimal places = $Pi2"
```