Σενάρια Κελύφους

Εργαστήριο Λειτουργικά Συστήματα

Σενάριο κελύφους (shell script) (1)

- Ένα αρχείο που περιέχει εντολές ονομάζεται σενάριο (script), ενώ αν περιέχει εντολές κελύφους ονομάζεται σενάριο κελύφους (shell script).
- Το κέλυφος μπορεί να διαβάσει εντολές εκτός από την κανονική είσοδο και από ένα αρχείο. Αν εκτελέσουμε μια εντολή κελύφους και δώσουμε ως όρισμα ένα όνομα αρχείου, τότε το κέλυφος αντί να διαβάσει εντολές από το τερματικό (κανονική είσοδο) θα τις διαβάσει από το αρχείο.
- Τα σενάρια κελύφους είναι αρχεία κειμένου τα οποία θα πρέπει να αρχίζουν πάντα με το πρόθεμα «#!» το οποίο ονομάζεται shebang.
- Το shebang έχει ιδιαίτερη σημασία μιας και δηλώνει ότι το συγκεκριμένο αρχείο είναι αρχείο σεναρίου.

Σενάριο κελύφους (shell script) (2)

- Μετά το shebang πρέπει να ορίζεται η *απόλυτη διαδρομή* προς τον διερμηνευτή.
  - Ένα σενάριο για το κέλυφος Bash πρέπει να αρχίζει με **#!/bin/bash**
  - Ενώ ένα σενάριο για το κέλυφος C-shell **#!/bin/csh**
- Μετά την ολοκλήρωση της σύνταξης ενός σεναρίου, θα πρέπει στο αρχείο να δοθεί η άδεια εκτέλεσης (μιας και εξ ορισμού το Unix/Linux για όλα τα αρχεία που δημιουργούνται δεν ορίζει την άδεια εκτέλεσης).
- Ένα αρχείο σεναρίου θα πρέπει υποχρεωτικά να έχει τις άδειες ανάγνωσης και εκτέλεσης, σε αντίθεση με τα δυαδικά εκτελέσιμα αρχεία (binary executables files), όπου αρκεί η άδεια εκτέλεσης.

Σενάριο κελύφους (shell script) (3)

Συνοπτικά ένα σενάριο κελύφους:

- θα πρέπει να αρχίζει με #! (χωρίς κενό διάστημα πριν και μετά).
- μετά το #! θα πρέπει να ορίζεται η απόλυτη διαδρομή προς το διερμηνευτή του κελύφους που χρησιμοποιούμε.
- θα πρέπει να έχει τις άδειες ανάγνωσης και εκτέλεσης.

Πως εκτελείται ένα σενάριο κελύφους

Για να εκτελέσουμε ένα σενάριο:

- Πρώτα θα πρέπει να δώσουμε την άδεια εκτέλεσης πληκτρολογώντας:

  π.χ. chmod u+x script1

Στη συνέχεια για να εκτελέσουμε το σενάριο πληκτρολογούμε:

- ./όνομα_αρχείου_σεναρίου π.χ. ./script1
- bash όνομα_αρχείου_σεναρίου π.χ. bash script1

Παράδειγμα σύνταξης script (1)

```
#!/bin/bash
echo "Please enter your name"
read name
echo "Please enter your age"
read age
echo "Please enter your sex. Male/Female"
read sex
echo "So you're a $age year old $sex called $name"
```

Παράδειγμα σύνταξης script (2)

```
#!/bin/bash
echo "Please give a sentence : "
read foo
var=$(echo $foo | tr "{a-z}" "{A-Z}")
echo $var
```

Αριθμητική κελύφους

- Τα κελύφη δεν υποστηρίζουν μεταβλητές αριθμητικού τύπου, όπως για παράδειγμα integer, float κλπ. Κατά συνέπεια δεν υποστηρίζουν αριθμητικές πράξεις.
- Το Bourne Again Shell (bash) υποστηρίζει πράξεις μόνο μεταξύ ακεραίων αριθμών, ενώ το Bourne Shell (sh) δεν υποστηρίζει καμία αριθμητική πράξη.
- Παρ' όλα αυτά υπάρχει τρόπος για την εκτέλεση αριθμητικών πράξεων σε κελύφη που δεν υποστηρίζουν αριθμητικές πράξεις, με τη χρήση εξωτερικών εντολών.
- Η πιο διαδεδομένη εντολή είναι η "expr".

Εντολή expr

- Η εντολή expr έχει τη δυνατότητα να πραγματοποιεί πράξεις μόνο μεταξύ ακεραίων αριθμών.
- Δέχεται σαν όρισμα μια αριθμητική παράσταση και τυπώνει στην έξοδο το αποτέλεσμα.
- Σαν ορίσματα στην expr μπορούν να δοθούν:
  - Σταθερές τιμές αριθμών.
  - Μεταβλητές στις οποίες προηγουμένως τους έχει ανάθεση μιας ακέραιας τιμής.
- Τα ορίσματα θα πρέπει να διαχωρίζονται με κενούς χαρακτήρες
- Η expr υποστηρίζει τις πράξεις: πρόσθεση, αφαίρεση, πολλαπλασιασμό, ακέραια διαίρεση, υπόλοιπο ακέραιας διαίρεσης.

Παραδείγματα εντολής expr (1)

Εντολή	Οθόνη
:~$ expr 2+3	**2+3 (λάθος λόγω έλλειψης κενών)**
:~$ expr 2 +3	**expr: syntax error: unexpected argument '+3'**
:~$ expr 2 + 3	**5**
:~$ a=10	
:~$ expr "$a" + 5	**15**
:~$ b=20	
:~$ expr "$a" + "$b"	**30**

Παραδείγματα εντολής expr (2)

Εντολή	Οθόνη	
:~$ expr 4 * 2	**8**	
:~$ expr 4 “*” 2	**8**	
:~$ expr 4 ‘*’ 2	**8**	
:~$ a=`expr 7 / 2`		
:~$ echo “$a”	**3**	**(όχι 3,5 !!!)**
:~$ b=2		
:~$ c=`expr “$a” + “$b” `		
:~$ echo “$c”	**5**	

Παραδείγματα εντολής expr (3)

Όταν χρησιμοποιείται το bash αντί των ανάποδων εισαγωγικών που μπορεί να μπερδέψουν τον αναγνώστη της εντολής, μπορεί να χρησιμοποιηθεί το $(). Στο παραδοσιακό sh δεν μπορεί να εφαρμοστεί αυτό.

Συνεπώς το προηγούμενο παράδειγμα μπορεί να συνταχθεί στο bash εξ' ίσου σωστά ως εξής:

Εντολή	Οθόνη
:~$ c=$(expr "$a" + "$b")	
:~$ echo "$c"	***5***

Αριθμητική κατάσταση του bash

- Το κέλυφος bash έχει ενσωματωμένη τη δυνατότητα εκτέλεσης αριθμητικών πράξεων που ονομάζεται «**αριθμητική κατάσταση**», η οποία είναι πιο ευέλικτη σε σχέση με την εντολή expr, διότι:
  - Δεν δημιουργείται πρόβλημα με τη χρήση ή όχι των κενών διαστημάτων.
  - Αν μια μεταβλητή δεν περιέχει αριθμό, θεωρείται ότι περιέχει τον αριθμό μηδέν.
  - Δεν δημιουργείται πρόβλημα να δεν έχει ανατεθεί τιμή σε μια μεταβλητή (αρκεί να μην γίνει χρήση του "$" πριν το όνομά της).
  - Η τιμή μιας παράστασης υπολογίζεται πιο γρήγορα, μιας και δεν δημιουργείται νέα διεργασία, αλλά η παράσταση ερμηνεύεται και υπολογίζεται από το ίδιο το κέλυφος.
- Η αριθμητική κατάσταση σηματοδοτείται με διπλές παρενθέσεις (( )), μέσα στις οποίες δεν επιτρέπεται η χρήση του "$" και υποστηρίζει όλες τις πράξεις.

Παραδείγματα αριθμητικής κατάστασης (1)

Εντολή	Οθόνη
:~$ a=$((3+2))	
:~$ echo "$a"	**5**
:~$ b=$((5*a))	
:~$ echo "$b"	**25**
:~$ c=$((a+d))	
**:~$** echo "$c"	**5** (εμφάνισε μόνο την τιμή της a γιατί η d δεν έχει τιμή)
:~$ c=$((a+$d)	
:~$ echo "$c"	**-bash a+: syntax error: operand expected**

Παραδείγματα αριθμητικής κατάστασης (2)

Εντολή	Οθόνη
:~$ a=10	
:~$ ((a++))	
:~$ echo "$a"	**11**
:~$ ((a*=2))	
:~$ echo "$a"	**22**
:~$ ((b=a+2))	
**:~$** echo "$b"	**24**
:~$ c=$((a+8))	
:~$ echo "$c"	**30**

Εντολή test (1)

- Η εντολή test χρησιμοποιείται συνήθως με την εντολή if για τον έλεγχο συνθηκών και συγκρίσεων μεταξύ:
  - Αρχείων
  - Συμβολοσειρών
  - Ακέραιων αριθμών
  - Λογικών τελεστών
- Συντάσσεται ως εξής: test expression ή [expression], όπου expression είναι μια συνθήκη, η οποία λαμβάνει την τιμή αληθής (true) ή ψευδής (false).
- Ο πίνακας που ακολουθεί περιέχει τις παραμέτρους της εντολής test που αφορούν σε αρχεία.

Εντολή test (2)

Expression	Is true if . . .
file1 -ef *file2*	*file1* and *file2* have the same inode numbers (the two filenames refer to the same file by hard linking).
file1 -nt *file2*	*file1* is newer than *file2*.
file1 -ot *file2*	*file1* is older than *file2*.
-b *file*	*file* exists and is a block-special (device) file.
-c *file*	*file* exists and is a character-special (device) file.
-d *file*	*file* exists and is a directory.
-e *file*	*file* exists.
-f *file*	*file* exists and is a regular file.
-g *file*	*file* exists and is set-group-ID.
-G *file*	*file* exists and is owned by the effective group ID.

Εντολή test (3)

Expression	Is true if . . .
`-k` *file*	*file* exists and has its "sticky bit" set.
`-L` *file*	*file* exists and is a symbolic link.
`-O` *file*	*file* exists and is owned by the effective user ID.
`-p` *file*	*file* exists and is a named pipe.
`-r` *file*	*file* exists and is readable (has readable permission for the effective user).
`-s` *file*	*file* exists and has a length greater than zero.
`-S` *file*	*file* exists and is a network socket.
`-t` *fd*	*fd* is a file descriptor directed to/from the terminal. This can be used to determine whether standard input/output/error is being redirected.
`-u` *file*	*file* exists and is setuid.
`-w` *file*	*file* exists and is writable (has write permission for the effective user).
`-x` *file*	*file* exists and is executable (has execute/search permission for the effective user).

Εντολή test (4)

Ο πίνακας που ακολουθεί περιέχει τις παραμέτρους της εντολής test που αφορούν σε συγκρίσεις αλφαριθμητικών.

Expression	Is true if . . .
string	*string* is not null.
-n *string*	The length of *string* is greater than zero.
-z *string*	The length of *string* is zero.
string1 = *string2* *string1* == *string2*	*string1* and *string2* are equal. Single or double equal signs may be used, but the use of double equal signs is greatly preferred.
string1 != *string2*	*string1* and *string2* are not equal.
string1 > *string2*	*string1* sorts after *string2*.
string1 < *string2*	*string1* sorts before *string2*.

Εντολή test (5)

Ο πίνακας που ακολουθεί περιέχει τις παραμέτρους της εντολής test που αφορούν σε συγκρίσεις ακέραιων αριθμών.

Expression	Is true if . . .
integer1 -eq *integer2*	*integer1* is equal to *integer2*.
integer1 -ne *integer2*	*integer1* is not equal to *integer2*.
integer1 -le *integer2*	*integer1* is less than or equal to *integer2*.
integer1 -lt *integer2*	*integer1* is less than *integer2*.
integer1 -ge *integer2*	*integer1* is greater than or equal to *integer2*.
integer1 -gt *integer2*	*integer1* is greater than *integer2*.

Εντολή test (6)

Ο πίνακας που ακολουθεί περιέχει τις παραμέτρους της εντολής test που αφορούν σε λογικούς τελεστές.

Operation	test	[[ ]] and (( ))
AND	-a	&&
OR	-o	||
NOT	!	!

Εντολή if - then

Η εντολή if μπορεί να συνταχθεί με δύο τρόπους, είτε με τη χρήση της εντολής test είτε χωρίς αυτή, ως εξής:

```
if test expression; then
        commands
fi
```

ή ισοδύναμα

```
if [ expression ]; then
        commands
fi
```

Παράδειγμα if - then

```
if test –f $file; then
        echo "$file is a regular file"
fi

echo "End of program."
```

Εντολή if - else

```
if test expression; then
        commands_1
else
        commands_2
fi
```

ή ισοδύναμα

```
if [ expression ]; then
        commands_1
else
        commands_2
fi
```

Παράδειγμα if - else

```
if test –f $file; then
        echo "$file is a regular file"
else
        echo "$file is not a regular file"
fi
echo "End of program."
```

Εντολή if - elif

```
if test expression1; then
        commands_1
elif   test expression2; then
        commands_2
else
        commands_3
fi
```

ή ισοδύναμα

```
if [ expression1 ]; then
        commands_1
elif    [ expression2 ]; then
        commands_2
else
        commands_3
fi
```

Παράδειγμα if - elif

```
if test –f $file; then
        echo "$file is a regular file"
elif test –d $file; then
        echo "$file is a directory"
else
        echo "$file is not a regular file or directory"
fi
echo "End of program."
```

Εντολή case

```
case value in
        pattern_1)  commands_1
                    ;;
        pattern_2)  commands_2
                    ;;
        .
        .
        .
         pattern_n)  commands_n
                    ;;
esac
```

Παράδειγμα case

```
case $day in
  "Δευτέρα" | "Τρίτη" | "Τετάρτη" | "Πέμπτη" | "Παρασκευή")
    echo "$day: Είναι εργάσιμη ημέρα."
    ;;
  "Σάββατο" | "Κυριακή")
    echo "$day: Είναι Σαββατοκύριακο."
    ;;
  *)
    echo "Παρακαλώ εισάγετε μια έγκυρη ημέρα της εβδομάδας."
    ;;
esac
```

Καλή συνέχεια…