"UNIX was not designed to stop its users from doing stupid things, as that would also stop them from doing clever things.

Εργαστήριο $5^ο$: Εισαγωγή στα Shell Scripts

Τσαδήμας Ανάργυρος, Τμήμα Πληροφορικής & Τηλεματικής

Τα shell scripts είναι αρχεία τα οποία περιέχουν εντολές, μεταβλητές και συνθήκες ελέγχου, τις οποίες αναλαμβάνει να εκτελέσει ο φλοιός.

Εισαγωγή στα shell scripts

Οι φλοιοί του Unix έχουν τη δυνατότητα να εκτελούν δέσμες εντολών. Μία δέσμη εντολών αποθηκεύεται σε ένα αρχείο και από εκεί υποβάλλεται προς εκτέλεση. Τα προγράμματα που συγγράφονται με τον τρόπο αυτό καλούνται προγράμματα φλοιού (shell scripts). Τα προγράμματα φλοιού μπορούν να περιέχουν, πέραν των κανονικών εντολών του Unix, και συγκεκριμένες δομές ελέγχου, που μας παρέχουν τη δυνατότητα για υπό συνθήκη εκτέλεση, επανάληψη κ.λπ. Η σύνταξη των δομών ελέγχου διαφέρει από φλοιό σε φλοιό.

Διαδικασία δημιουργίας και εκτέλεσης ενός shell script

Για να δημιουργήσουμε και να εκτελέσουμε ένα αρχείο το οποίο περιέχει εντολές, θα πρέπει να ακολουθήσουμε τα εξής βήματα:

1. Δημιουργούμε το αρχείο και γράφουμε μέσα σε αυτό τις εντολές που θέλουμε να εκτελεστούν π.χ. δημιουργούμε το αρχείο `myscript`
2. Κάνουμε το αρχείο εκτελέσιμο `chmod +x myscript`
3. Εκτελούμε το αρχείο
   - αν είμαστε στον ίδιο κατάλογο με το αρχείο εκτελούμε `./myscript`
   - αν το αρχείο είναι σε έναν κατάλογο που έχει δηλωθεί στη μεταβλητή PATH εκτελούμε `myscript`
   - μπορούμε να εκτελέσουμε το αρχείο χρησιμοποιώντας κάποιο συγκεκριμένο φλοιό καλώντας πρώτα το όνομα φλοιού και δίνοντας όρισμα το όνομα του εκτελέσιμου αρχείου π.χ. `csh myscript`

Στην πρώτη γραμμή του script file μπορούμε να ορίσουμε τον φλοιό σύμφωνα με τον οποίο θα εκτελεστεί. Αν δεν οριστεί θεωρείται ότι θα υποβληθεί στο φλοιό sh. Ο ορισμός γίνεται χρησιμοποιώντας τα σύμβολα #! ακολουθούμενα από το path του φλοιού π.χ. #!/bin/bash [1]

Για να προσθέσουμε σχόλια στο script χρησιμοποιούμε το σύμβολο #. Οτιδήποτε υπάρχει μετά από αυτό το σύμβολο και μέχρι το τέλος της γραμμής αγνοείται από τον φλοιό. ΜΕ ΜΙΑ ΕΞΑΙΡΕΣΗ: όταν η γραμμή ξεκινάει με #! είναι ειδική περίπτωση όπως είπαμε πιο πάνω (Κεφάλαιο 11 του βιβλίου Linux command line and shell scripting Bible[1]).

[1] Δείτε παράδειγμα εδώ τι γίνεται με τα κενά και τους ειδικους χαρακτήρες στα scipt

Μέχρι τώρα τα scripts εκτελούνται χρησιμοποιώντας έναν υποφλοιό, ο οποίος εκκινείται, εκτελεί το αρχείο και πεθαίνει. Στην περίπτωση όμως που θέλουμε να κάνουμε αλλαγές στο αρχείο .profile και θέλουμε να προσθέσουμε νέες μεταβλητές περιβάλλοντος ή aliases και θέλουμε να τα δοκιμάσουμε χωρίς να χρειαστεί να κάνουμε logout μπορούμε να εκτελέσουμε το script, δηλαδή το αρχείο .profile στον τρέχοντα φλοιό ως εξής

```
. .profile
```

Η τελεία είναι μια εντολή που παίρνει σαν όρισμα ένα όνομα αρχείου και το εκτελεί στον τρέχοντα φλοιό. Οποιαδήποτε αλλαγή στο τρέχον περιβάλλον θα παραμείνει μετά την εκτέλεση του script (όταν χρησιμοποιούμε την τελεία, δεν χρειάζεται το αρχείο να είναι εκτελέσιμο, αρκεί μόνο να μπορούμε να το κάνουμε read[2]).

Περνώντας ορίσματα στα scripts

Μπορούμε να περάσουμε command-line ορίσματα στα scripts. Όταν εκτελούμε ένα script, ειδικές μεταβλητές του φλοιού ρυθμίζονται αυτόματα προκειμένου να ταιριάξουν τα ορίσματα. Αυτές οι μεταβλητές είναι γνωστές ως παράμετροι θέσης (positional parameters). Οι παράμετροι $1, $2, $3 μέχρι το $9 αναφέρονται στο πρώτο, δεύτερο, τρίτο κτλ όρισμα που θα δοθεί στο command line [3]. Η παράμετρος $0 κρατά το όνομα του shell script. Αν θέλουμε να μετρήσουμε πόσα ορίσματα έχουν περάσει στο script μπορούμε να χρησιμοποιήσουμε τη μεταβλητή $#. Τέλος, αν θέλουμε να περάσουμε στο script όλα τα ορίσματα που θα δοθούν από το command line μπορούμε να χρησιμοποιήσουμε την $*. Η $* επιστρέφει όλα τα ορίσματα που δόθηκαν χωρισμένα με ένα κενό μεταξύ τους. Επιπλέον η $@ φυλάει όλες τις παραμέτρους αλλά μπορεί να τις ξεχωρίσει αν έχουν κενά. Παράδειγμα[4]:

```
echo Running Program $0
echo The first three arguments are:
echo $1
echo $2
echo $3
echo Here are all $# arguments:
echo $*
```

Αριθμητικοί τελεστές

Αν δοκιμάσετε να αναθέσετε αριθμητικές τιμές σε μεταβλητές και να κάνετε πράξεις π.χ.

```
x=2
x=$x+2
echo $x
```

Θα περιμένατε να δείτε το 3, αλλά θα δείτε 2+1. Στην ουσία έγινε ένωση δύο αλφαριθμητικών, του “2” και του “+1”. Για να εκτελέσουμε αριθμητικούς τελεστές χρειαζόμαστε την εντολή expr. Χρειάζεται προσοχή όμως γιατί αν δεν μπουν κενά πριν και μετά τον αριθμητικό τελεστή η expr δεν θα ερμηνεύσει την αριθμητική έκφραση. Επίσης ο πολλαπλασιαμός γίνεται χρησιμοποιώντας το *. Επίσης δεν μπορούμε να χρησιμοποιήσουμε π.χ. εκφράσεις με δυνάμεις. Οι τελεστές της expr[5] εμφανίζονται στον πίνακα 1. Δείτε παραδείγματα στο http://linux.101hacks.com/unix/expr/ και στο https://goo.gl/Q4kaWQ.

Για να καταχωρήσουμε το αποτέλεσμα μιας εντολής expr σε μια μεταβλητή χρησιμοποιούμε τα ‘ ‘[6] π.χ.
x = ‘expr $x + 1‘.
Υπάρχει και η εντολή let η οποία είναι πιο απλή και χρησιμοποιεί την αριθμητική τιμή μιας μεταβλητής απευθείας χωρίς το σύμβολο $ π.χ.

```
x=100
let y=2*(x+5)
```

[2]Για να έχουμε auto-completion με τη read και να εμφανίσουμε και μήνυμα στο χρήστη, χρησιμοποιούμε την read -e -p "Give a parameter"

[3]Για να αναφερθούμε σε περισσότερα ορίσματα χρησιμοποιούμε αγκύλες π.χ, ${12} αναφέρεται στο 12-ο όρισμα

[4]δείτε μια άλλη εκδοχή εδώ

[5]Προσοχή: Κάποιες από τις εκφράσεις δεν δουλεύουν στο Solaris όπως η substr, length, ενώ η let αντί για παρενθέσεις θέλει \(, \)

[6]Αυτό λέγεται command substitution

Τελεστής	Περιγραφή
*	πολλαπλασιασμός
/	διαίρεση
%	υπόλοιπο
\&	λογικό και (and)
|	λογικό ή (or)
=,!=	ίσο,διάφορο
=\>, =\<	μεγαλύτερο ή ίσο, μικρότερο ή ίσο
\>, \<	μεγαλύτερο, μικρότερο
substr (str) (start) (lenhth)	αντικατάσταση του str που ξεκινά από τη θέση start και εχει length χαρακτηρες
index (str) (charlist)	η θέση του πρώτου χαρακτήρα του str που εμφανίζεται το charlist
length (str)	το μήκος του str
\(, \)	παρενθέσεις

Πίνακας 1: τελεστές της expr

```
echo $y
```

Θα μας επιστρέψει 210. Αντί για τη let μπορεί να χρησιμοποιήσουμε διπλές παρενθέσεις. π.χ. η let x=x+3 μπορεί να γραφεί και ως ((x= x+3))

Άσκηση: Φτιάξτε ένα shell script το οποίο να προσθέτει 2 αριθμούς.

η εντολή exec

Η εντολή exec αντικαθιστά την τρέχουσα διεργασία φλοιού με την καθορισμένη εντολή που δίνουμε σαν όρισμα. Συνήθως, όταν εκτελούμε μια εντολή στο φλοιό, αυτή προκύπτει από την κλωνοποίηση του φλοιού και τη διαφοροποίηση του κώδικα εντολών του, ώστε να προκύψει μια νέα θυγατρική διεργασία. Στην περίπτωση της exec αντικαθίσταται ο κώδικας του φλοιού με αυτόν της εντολής. Υλοποιεί την κλήση συστήματος exec του unix.

Άσκηση: Αλλάξτε τον τρέχοντα φλοιό σας σε csh.

0.0.1 Σημείωση για τις ανακατευθύνσεις

Εκτός από τον απλή ανακατεύθυνση εισόδου (<) υπάρχει και η inline ανακατεύθυνση εισόδου ή αλλιώς here document. Συμβολίζεται με « και παίρνει σαν όρισμα ένα delimeter, το οποίο πρέπει να είναι μια λέξη η οποία δηλώνει την αρχή και το τέλος του input το οποίο μπορέι να αποτελείται από πολλές γραμμές. Παράδειγμα:

```
$wc << EOF
> test string 1
> test string 2
> test string 3
> EOF
2 9 42
$
```

Δείτε περισσότερα στο https://goo.gl/1yPgBV.

Επίσης, μπορούμε να ανακατευθύνουμε την έξοδο στο πουθενά, βάζοντας την ανακατεύθυνση εξόδου και το ειδικό αρχείο /dev/null, δηλαδή

```
command > /dev/null
```

0.0.2 Σημείωση για τα flags του bash

Με την ειδική μεταβλητή $- μπορούμε να δούμε ποια flags είναι ενεγρά στον bash που είμαστε συνδεδεμένοι. Τα options ή flags είναι ρυθμίσεις που μας επιτρέπουν να αλλάξουμε τη συμπεριφορά του φλοιού ή ενός shell script. Ρυθμίζονται με την εντολή set.

```
echo $-
himBH
```

- h σημαίνει να θυμάται ο φλοιός τη θέση των εντολών (εκτελέσιμων αρχείων) καθώς αναζητάται η εντολή για εκτέλεση.
- i σημαίνει "interactive"
- m σημαίνει "monitor" δηλαδή job control
- B σημαίνει "brace expand"

- Η σημαίνει "history expand". Με αυτό μπορούμε να τρέξουμε μια εντολή από το history με τον αύξοντα αριθμό της.

Ειδικά αρχεία στο UNIX (Pseudo-devices)

- /dev/null ➜ λαμβάνει είσοδο και την απποριπτει. Δεν παράγει έξοδο. Επιστρέφει end-of-file σε ανάγνωση
- /dev/zero ➜ λαμβάνει είσοδο και την απορρριπτει. Επιστρέφει ένα συνεχές ρεύμα από NUL (zero value) bytes.
- /dev/full ➜ παράγει ένα συνεχές ρεύμα από NUL (zero value) bytes όταν διαβάζεται και επιστρέφει ένα μήνυμα "disk full" όταν γράφουμε σε αυτό.
- /dev/random ➜ παράγει ένα μεταβλητού μήκους ψευδο-τυχαίων αριθμών.

https://goo.gl/EoQuWt.

Δομή ενός shell script

Πάντα θα πρέπει να έχουμε κάτι σαν μικρό manual για το script μας. Σε περίπτωση που ζητάμε ορίσματα από τον χρήστη καλό είναι να τον ενημερώνουμε για το τι ακριβώς ζητάμε. Αυτό γίνεται συνήθως τυπώνοντας ένα απλο "Usage: script_name parameter1 parameter2".

Για να είμαστε σίγουροι ότι το script μας θα εκτελεστεί σωστά, θα πρέπει να γίνονται πάντα κάποιοι έλεγχοι πριν την εκτέλεση του κυρίως script. Μερικά πράγματα που έχει νόημα να ελέγξουμε είναι:

- Αν υπάρχουν τα αρχεία στα οποία θα επιδράσουμε
- Αν υπάρχουν στο σύστημα οι εντολές που θέλουμε να εκτελεστούν ή αν είναι στο σωστό μέρος
- οτιδήποτε άλλο θα μπορούσε να έχει πειραχτεί ή να μην ισχύει στο σύστημα στο οποίο θα εκτελετεί το script. π.χ. να μην είναι εκτελέσιμο ένα αρχείο το οποίο ζητάμε να εκτελέσουμε.

Όλοι αυτοί οι έλεγχοι ονομάζονται sanity checks. Συνήθως μπαίνουν στην αρχή ενός shell script για να διασφαλίσουν την ορθή λειτουργία του.

Στο τέλος ενός shell script πρέπει να λαμβάνουμε υπ'όψιν μας ότι πρέπει να κάνουμε cleanup. Για παράδειγμα αν έχουμε χρησιμοποιήσει κάποια προσωρινά αρχεία, να τα σβήσουμε. Επίσης καλό είναι να δίνουμε και έναν κωδικό εξόδου (exit 0) ώστε να ξέρουμε αν όλα πήγαν καλά και η εκτέλεση του script έγινε κανονικά. Δείτε στο https://goo.gl/w1q2Sh κάποιους κανόνες.

Δομές εκτέλεσης υπό συνθήκη

Συνθήκες

Στο Unix δεν υπάρχει ξεχωριστή έννοια συνθήκης όπως σε άλλες γλώσσες. Οποιαδήποτε εντολή μπορεί να χρησιμοποιηθεί σαν συνθήκη. Για παράδειγμα μια εντολή read μπορεί να είναι συνθήκη για κάποιον βρόχο. Εφόσον η εντολή εκτελείται σωστά επιστρέφει κωδικό σωστής εκτέλεσης και η «συνθήκη» επαληθεύεται. Εφόσον εκτελεστεί λάθος επιστρέφει κωδικό λάθους και η συνθήκη είναι ψευδής. Ο κωδικός αυτός στο UNIX επιστρέφεται στην παράμετρο $? και για σωστή εκτέλεση είναι ο αριθμός 0 ενώ για λάθος εκτέλεση είναι ένας θετικός ακέραιος αριθμός (δείτε τον πίνακα 2). Ορισμένες εντολές επιστρέφουν περισσότερους κώδικες για λάθος εκτέλεση όπως πχ η εντολή expr.

Οι συνθήκες συντάσσονται ως εξής: [συνθήκη] [7] ή test συνθήκη. Η test επιστρέφει 0 αν η συνθήκη είναι αληθής. Η έξοδος της test χρησιμοποιείται από δομές ελέγχου. Προσοχή κατά την πρώτη σύνταξη να παρεμβάλλεται κενό ανάμεσα στη συνθήκη και τις αγκύλες. Για μια ολοκληρωμένη λίστα των file expressions δείτε εδώ. Μερικοί

[7] δείτε εδώ κι εδώ

Κωδικός	Περιγραφή
0	Επιτυχής ολοκλήρωση της εντολής
1	Γενικό άγνωστο σφάλμα
2	Κακή χρήση εντολής φλοιού
126	Η εντολή δεν μπορεί να εκτελεστεί
127	Η εντολή δεν βρέθηκε
128	Μη έγκυρο όρισμα εξόδου
128+x	Σφάλμα με το σήμα x
130	Τερματισμός εξαιτίας Ctrl+C
255	Κωδικός εξόδου εκτός ορίων

Πίνακας 2: Κωδικοί εξόδου εντολών στο Linux

από τους ελέγχους που μπορούμε να κάνουμε είναι:

- Σε αρχεία (-r, -w, -x, -f, -d, -s)
  - -r filename : true αν το filename υπάρχει σαν readable αρχείο.
  - -w filename : true αν το filename υπάρχει σαν writeable αρχείο.
  - -x filename : true αν το filename υπάρχει σαν executable αρχείο.
  - -f filename : true αν το filename υπάρχει σαν αρχείο (όχι κατάλογος).
  - -d directoryname : true αν το filename υπάρχει σαν κατάλογος.
  - -s filename : true αν το filename υπάρχει και δεν είναι κενό.
- Σε συμβολοσειρές (-z, -n, -l, =, !=)
  - -z stringname : true αν το string είναι κενό.
  - -n stringname : true αν το string δεν είναι κενό.
  - str1=str2 : true αν το str1 είναι ισο με το str2.
  - str1!=str2 : true αν το str1 είναι διαφορετικό από το str2.
- Λογικοί τελεστές (!, -a, -o)
  - ! expression : true αν η expression είναι false.
  - expr1 -a expr2 : true αν και η expr1 και η expr2 είναι αληθείς.
  - expr1 -o expr2 : true αν η expr1 ή η expr2 είναι αληθείς.
- Σε integers
  - n1 -eq n2 : το n1 ισούται με το n2
  - n1 -ne n2 : το n1 δεν ισούται με το n2
  - n1 -gt n2 : το n1 είναι μεγαλύτερο από το n2
  - n1 -ge n2 : το n1 είναι μεγαλύτερο ή ίσο από το n2
  - n1 -lt n2 : το n1 είναι μικρότερο από το n2
  - n1 -le n2 : το n1 είναι μικρότερο ή ίσο από το n2

Προσοχή στις συγκρίσεις!

Η test χρησιμοποιεί τα μαθηματικά σύμβολα για να συγκρίνει strings και αλφαρηθμιτικές εκφράσεις για να συγκρίνει αριθμούς. Αν δεν τα χρησιμοποιήσετε με αυτή τη λογική θα έχετε πρόβλημα!

Χρήση εισαγωγικών [8]

Χρήση εισαγωγικών Τα απλά εισαγωγικά ('single quotes') άρουν την μεταχαρακτηρική ιδιότητα όλων των συμβόλων εκτός από τον εαυτό τους. Τα διπλά εισαγωγικά ("double quotes") άρουν την μεταχαρακτηρική ιδιότητα όλων των συμβόλων εκτός από τον εαυτό τους, την ανάποδη κάθετο (backslash) και το δολάριο ($). Τα ανάποδα εισαγωγικά ('back quotes') προκαλούν την εκτέλεση της εντολής που περικλείουν.

Η δομή ελέγχου if[9]

Η εντολή if εκτελεί ένα, δύο ή περισσότερες εναλλακτικές σειρές εντολών βασιζόμενη στο status code μιας εντολής ή συνδυασμού εντολών.

```
if [expression1]
then
   command-set-1
elif [expression2]
then
```

[8] https://goo.gl/e9Nj9m

[9] Παρατήρηση: Αν βάλετε μια έκφραση π.χ. την expression1 στην ίδια γραμμή με το then, πρέπει να ξεχωρίσετε τη συνθήκη με το then με το semicolon (;).

```
    command-set-2
else
    command-set-3
fi
```

Παράδειγμα ενός script για πρόσθεση δυο αριθμών

```
#!/bin/bash
if [ $# -eq 2 ]
then
    echo  expr $1 + $2
else
    echo "usage: $0 number1 number2"
fi
```

Αναφορές

[1] R. Blum. Linux command line and shell scripting bible, volume 481. John Wiley & Sons, 2008.