Λειτουργικά Συστήματα Ι

Ασκήσεις Πράξης

Τμήμα Μηχανικών Πληροφορικής & Υπολογιστών

Σχολή Μηχανικών

Περισσότερα για τον Φλοιό

Περισσότερα για μεταβλητές

Το μήκος μιας μεταβλητής δίνεται από την παράσταση **${#**VARIABLE**}**. Θυμίζουμε πως με παρόμοιο τρόπο αναφερόμαστε και στο πλήθος των στοιχείων ενός πίνακα.

```
$ echo $HOME
/home/nemo
$ echo ${#HOME}
10
$ USERS=(Alice Bob Carol Dave)
$ echo ${#USERS[@]}
4
```

Περισσότερα για μεταβλητές

Ανάθεση προκαθορισμένης τιμής σε μεταβλητής στην περίπτωση που δεν έχει τεθεί: **${**VARIABLE**:-**defaultValue**}**
Η προκαθορισμένη τιμή δεν αποθηκεύεται στη μεταβλητή.

```
$ BACKUPDIR=/backup; echo $BACKUPDIR
/backup
$ COPYTO=${BACKUPDIR:-/tmp}; echo $COPYTO
/backup
$ COPYTO=${NEWBACKUPDIR:-/tmp}; echo $COPYTO
/tmp
$ echo ${USERNAME:-`whoami`}
nemo
```

Περισσότερα για μεταβλητές

Ανάθεση προκαθορισμένης τιμής σε μεταβλητής στην περίπτωση που δεν έχει τεθεί: **${**VARIABLE**:=**defaultValue**}**
Η προκαθορισμένη τιμή αποθηκεύεται στη μεταβλητή.

```
$ MYVAR=${YOURVAR:=somevalue}     # :=
$ echo $MYVAR
somevalue
$ echo $YOURVAR
somevalue
$ MYVAR=${XYZ:-anothervalue}      # :-
$ echo $XYZ                       # XYZ is empty
```

Περισσότερα για μεταβλητές

Μήνυμα λάθους σε περίπτωση που δεν έχει τεθεί μία μεταβλητή: **${**VARIABLE**?**ERRMSG**}** **${**VARIABLE**:?**ERRMSG**}**
Η μορφή με "**:**" θα προκαλέσει λάθος αν η μεταβλητή δεν έχει τεθεί ή είναι κενή, ενώ η μορφή χωρίς "**:**" θα προκαλέσει λάθος μόνο αν η μεταβλητή δεν έχει τεθεί. Σε περίπτωση λάθους εμφανίζεται το μήνυμα ERRMSG και η τιμή εξόδου της εντολής παίρνει την τιμή 1.

```
$ VAR=${NEWVAR?Error: variable not exist}
bash: NEWVAR: Error: variable not exist
$ OLDVAR=""                          # set OLDVAR
$ VAR=${OLDVAR?Error: variable not exist}
```

Περισσότερα για μεταβλητές

```
function makeDir(){
  local DIR="$1"          # get directory name
  local PERM=${2:-0755}   # get permissions
  [ $# -eq 0 ] && { echo "$0: directory"; return 1; }
  [ ! -d "$DIR" ] && mkdir -m "$PERM" -p "$DIR"
}
makeDir "html/js" 0644
makeDir "html/jquery"

$ ./makeDirectories.sh; ls -l html
drwxr-xr-x 2 nemo staff 4096 Dec 12 04:33 jquery
drw-r--r-- 2 nemo staff 4096 Dec 12 04:33 js
```

Περισσότερα για μεταβλητές

Η παράσταση **${**VARIABLE**:**start**:**length**}** επιτρέπει τη διαμέριση (slicing) μιας μεταβλητής. Μπορούμε δηλαδή να εξάγουμε ένα τμήμα της. Αν το παραληφθεί το *length* θα εξάγουμε το τμήμα από *start* έως το τέλος της.

```
$ VAR=abcdefghijklmopqrstuvwxyz
$ echo ${VAR:5}
fghijklmopqrstuvwxyz
$ echo ${VAR:5:10}
fghijklmop
$ NEWVAR=${VAR:(-3)}; echo $NEWVAR
xyz
```

Περισσότερα για μεταβλητές

Να μετονομαστούν όλα τα αρχεία της Python του τρέχοντος καταλόγου των οποίων το όνομα αρχίζει με “14-”. Το νέο όνομα κάθε αρχείου θα αρχίζει από “15-” και το υπόλοιπο τμήμα θα παραμένει το ίδιο.

```
$ ls 14-*
14-01.py  14-02.py  14-xxxxx.py  14-xyz.py
$ for f in 14-*.py; do mv $f 15-${f:3}; done
$ ls 14-* 15-*
15-01.py  15-02.py  15-xxxxx.py  15-xyz.py
```

Περισσότερα για μεταβλητές

Η διαμέριση γίνεται και σε πίνακα:

```
$ A=(Alice Bob "Carol Li" Dave); echo ${#A[@]}
4
$ B=(${A[@]:2}); echo ${B[@]}
Carol Li Dave
$ C=(${A[@]:1}); echo ${C[@]}
Bob Carol Li Dave
$ echo ${C[@]: -2}
Li Dave
$ echo ${#C[@]}
4
```

Περισσότερα για μεταβλητές

Αφαίρεση τμήματος από τα αριστερά **${**VARIABLE**#**Pattern**}** και από τα δεξιά **${**VARIABLE**%**Pattern**}** μιας μεταβλητής:

```
$ PLAYNOW=/music/RHCP/Snow.mp3
$ TEMP=${PLAYNOW#/music/}
$ echo $TEMP
RHCP/Snow.mp3
$ ARTIST=${TEMP%/Snow.mp3}
$ echo $ARTIST
RHCP
$ ARTIST=${${PLAYNOW#/music/}%/Snow.mp3}
bash: ARTIST=${…}: bad substitution
```

Περισσότερα για μεταβλητές

Αντικατάσταση τμήματος μεταβλητής:

${VARIABLE**/**Pattern**/**Replacement**}** # first occurrence

${VARIABLE**//**Pattern**/**Replacement**}** # all occurrences

```
$ OS="I prefer Windows. I am a Windows fan."
$ OS=${OS/Windows/Linux}
$ echo $OS
I prefer Linux. I am a Windows fan.
$ OS="I prefer Windows. I am a Windows fan."
$ OS=${OS//Windows/Linux}
$ echo $OS
I prefer Linux. I am a Linux fan.
```

Περισσότερα για μεταβλητές (sed)

```
$ OS="I prefer Windows. I am a Windows fan."
$ echo $OS | sed 's/Windows/Linux/g'  # global
I prefer Linux. I am a Linux fan.
$ OS=$(echo $OS | sed 's/Windows/Linux/g')
$ echo $OS
I prefer Linux. I am a Linux fan.
$ OS=`echo $OS | sed 's/Windows/Linux/g'`
$ OS=$(sed 's/windows/Linux/g' <<< $OS)    #😱
```

Περισσότερα για μεταβλητές

- Μετατροπή του πρώτου χαρακτήρα σε κεφαλαίο με τον τελεστή “**^**” και όλων των χαρακτήρων με τον τελεστή “**^^**”.
- Μετατροπή του πρώτου χαρακτήρα σε πεζό με τον τελεστή “**,**” και όλων των χαρακτήρων με τον τελεστή “**,,**”.
- Ο τελεστής “**~**” μετατρέπει τον πρώτο χαρακτήρα από πεζό σε κεφαλαίο και το αντίστροφο και ο τελεστής “**~~**” όλους τους χαρακτήρες.

Περισσότερα για μεταβλητές

```
$ echo "${BASH_VERSION}
5.1.16(1)-release                                  # > 4.x
$ title="MSc in Machine learning"
$ echo $title
MSc in Machine learning
$ echo ${title^} - ${title^^}
MSc in Machine learning - MSC IN MACHINE LEARNING
$ echo ${title,} - ${title,,}
mSc in Machine learning - msc in machine learning
$ echo ${title~} - ${title~~}
mSc in Machine learning - msC IN mACHINE LEARNING
```

heredoc

Ένα “here document” μας παρέχει τη δυνατότητα να τροφοδοτήσουμε ένα πρόγραμμα με δεδομένα από την προκαθορισμένη είσοδο χωρίς να χρειάζεται να χρησιμοποιήσουμε κάποιο αρχείο.

Η γενική μορφή είναι:

command **<<WORD**
data
more data
and more data
WORD

```
$ tr a-z A-Z <<END_TEXT
> City of
> Black & White
> END_TEXT
CITY OF
BLACK & WHITE
```

heredoc

Αν το κείμενο με το οποίο τροφοδοτούμε το πρόγραμμα περιέχει μεταβλητή, θα προηγηθεί επέκταση μεταβλητής. Αν δεν το επιθυμούμε, θα πρέπει να περικλείσουμε το λεκτικό σε μονά εισαγωγικά:

```
$ OS="GNU Linux"
$ wc -w <<EOT
> I prefer
> $OS
> EOT
4
```

```
$ OS="GNU Linux"
$ wc -w <<'EOT'
> I prefer
> $OS
> EOT
3
```

here string

Ένα “here string” έχει παρόμοιο συντακτικό (<<< αντί <<) και λειτουργία και τροφοδοτεί ένα πρόγραμμα με τα περιεχόμενα ενός αλφαριθμητικού αντί για αρχείο.

```
$ tr a-z A-Z <<< 'one two three'
ONE TWO THREE
$ VAR='four five six'
$ tr a-z A-Z <<< "$VAR"
FOUR FIVE SIX
$ tr a-z A-Z <<< '$variable'
$VARIABLE
```

Ακολουθίες

Οι νεότερες εκδόσεις του φλοιού Bash διαθέτουν ενσωματωμένη υποστήριξη για τη δημιουργία αριθμητικών κυρίως ακολουθιών. Η παράσταση **{start..stop..step}** επιστρέφει μία ακολουθία από *start* έως *stop* με βήμα *step*, όπως δείχνουν τα παραδείγματα που ακολουθούν:

```
$ echo {1..5}
1 2 3 4 5
$ echo {1..20..2}
1 3 5 7 9 11 13 15 17 19
$ echo {a..z}
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

Ακολουθίες

```
$ echo A{1..5}B
A1B A2B A3B A4B A5B
$ echo x{X{1..4},Y{5..8}}y
xX1y xX2y xX3y xX4y xY5y xY6y xY7y xY8y
$ mkdir {2018..2020}{01..12}
$ ls -d 2*
201801  201802  201803  201804  201805  ...
202007  202008  202009  202010  202011  202012
$ echo File-{A,B,01,02}.txt
File-A.txt File-B.txt File-01.txt File-02.txt
```

Ακολουθίες

```
#!/bin/bash
# squaresv2.sh - for with brace expansion
for i in {1..10}              # for i in $(seq 10)
do
  echo -n "$(( i * i )) "
done
echo

$ chmod u+x squaresv2.sh
$ ./squaresv2.sh
1 4 9 16 25 36 49 64 81 100
```

Ερωτήσεις