

bash(1) scripting

Μιχάλης Ιατρού
iatrou@hellug.gr

Documentation

- Advanced Bash-Scripting Guide

<http://tldp.org/LDP/abs/html/index.html>

- `$ man 1 bash`
- `$ info bash`

Γιατί bash scripting;

- Απλό, εύκολη/γρήγορη εκμάθηση
- Ευρεία χρήση
 - System administration
 - Programming
- Διαρκής ανάπτυξη
- Advanced features
- Portability

Πότε λέμε όχι;

- Resource-intensive/heavy-weight processing tasks
- Math operations (floating point, arbitrary precision operations, complex numbers)
- Mission critical applications (security, availability, robustness)
- Heavy filesystem IO operations

Μεταβλητές

```
# Variables in bash are untyped
```

```
$ i=1
```

```
$ let $((i+=2))      # i = 3
```

```
$ i=${i/3/stringaki} # i = "stringaki"
```

```
$ i=${i/string/4}    # i = "4aki"
```

```
$ i=${i/aki/2}       # i = "42"
```

```
$ let "i = i ** 2"   # i = 1764
```

Ειδικοί τύποι μεταβλητών

- local
- environmental
- positional parameters
 - \$0, \$1, \$2 ... \${10}, \${11}
 - \$*, @\$
- Άλλες ειδικές μεταβλητές
 - \$#
 - \$?
 - \$\$
 - \$UID, \$RANDOM, \$SECONDS, ...

Χειρισμός μεταβλητών

- `${var:-foo}`
Use `var=foo` iff `var` unset
- `${var:+foo}`
Use `var=foo` iff `var` set
- `${var:=foo}`
Set `var=foo` iff `var` unset
- `${var:?foo}`
Print “foo” and `exit` iff `var` unset

Arrays

```
$ a[7]=42 # Zero-based indexing
$ echo ${a[7]}
$ a=(123 foo [3]=42)
$ echo ${#a[@]} # 4, Num of elements in array
$ echo ${#a[1]} # 3, Length of first element
$ echo ${a[1]/foo/bar} # "bar"
$ echo $a # 123, First element
```

Quoting

- `'...'` Full quoting, every special character gets interpreted literally
- `"..."` Partial quoting, escape sequences, variable referencing, command substitution
- `$'...'` Like `'...'` but interprets escape sequences

Partial quoting για μεταβλητές

- Είναι καλή πρακτική οι μεταβλητές να “προστατεύονται” με partial quoting
 - Τα λάθη από overquoting ανιχνεύονται εύκολα
 - Τα λάθη από ελλιπές quoting ανιχνεύονται δύσκολα
- Εξαιρέσεις
 - if [...]
 - if ((...))
 - Μεταβλητές τύπου integer (declare -i)

Test constructs

- if *command*
- if test ...
- if [...]
- if [[...]]
- if ((...))

if

if condition; then statements; fi

*if condition; then statements; elif condition;
then statements; else statements; fi*

```
if (( $UID == 0 )); then  
    echo "Using root is dangerous..."  
    rm -rf /  
fi
```

Tests

- Arithmetic comparison
 - gt, -lt, -eq, -ne, -le, -ge
- String comparison
 - =, ==, !=, > (\>), < (\<), -z, -n
- File tests
 - -e (exists), -f (regular file), -d (directory), -L (link), -r (read), -w (write), -x (execute)
 - f1 -nt f2 (f1 newer than f2), f1 -ot f2
- and, or statements
 - expr1 -a expr2, expr1 -o expr2

Loops: for

```
for x [in list]; do commands; done
```

- ```
for i in $(seq 1 4)
do
 cat /dev/urandom > /dev/sda${i}
done
```
- ```
for f in passwd shadow; do
    rm -f /etc/${f}
    echo "You need to reboot for changes to take
    effect"
done
```

Loops: while

```
while condition; do commands; done
```

- while :

```
do
```

```
    echo "All work and no play makes Jack a dull  
    boy"
```

```
done
```

- gawk -F: '{print \$(NF-1)}' /etc/passwd | sort -rd |

```
    uniq | while read H; do
```

```
        rm -rf $H
```

```
done
```

substrings

```
declare str="A simple string for practice"
```

```
echo "${str:0:8}" # A simple
```

```
echo "${str:2:6}" # simple
```

```
echo "${str: -8:8}" # practice
```

```
echo "${str: -12}" # for practice
```

Strings and patterns

- `${str#pattern}`

Remove from `str` the shortest part of `pattern` that matches the front end of `str`

- `${str##pattern}`

Remove from `str` the longest part of `pattern` that matches the front end of `str`

- `${str%pattern}`

Remove from `str` the shortest part of `pattern` that matches the back end of `str`

- `${str%%pattern}`

Remove from `str` the longest part of `pattern` that matches the back end of `str`

Προσοχή

```
cp $file1 $file2           # λάθος  
cp "$file1" "$file2"      # λιγότερο λάθος  
cp -- "$file1" "$file2"   # σωστό  
  
for f in `ls *.{1,2,4}`; do ... done #  
    λάθος  
for f in *.{1,2,4}; do ... done # σωστό
```

Προσοχή (2)

```
cat file | sed s/foo/bar/ > file # λάθος
```

```
cat file | sed s/foo/bar/ > tmp && mv tmp  
file # σωστό
```

```
sed -i s/foo/bar/ file # σωστό
```

```
echo $foo # λάθος
```

```
echo "$foo" # σωστό
```

Προσοχή (3)

```
for i in {1..10}; do ./foo & done # λάθος
```

```
for i in {1..10}; do ./foo & done # σωστό
```

```
$ echo "Hello World!" # λάθος
```

```
$ echo 'Hello World!' # σωστό
```

```
for arg in $* # λάθος
```

```
for arg in "$@" # σωστό
```

Ευχαριστώ ;-)