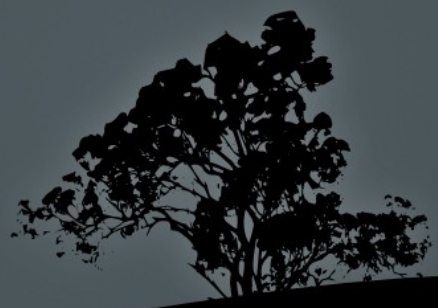
Εξέταση 102 – Μάθημα 2

105.2 Προσαρμογή ή συγγραφή απλών σεναρίων (scripts)



Δημιουργία σεναρίων (scripts)

- **$ cat > many.sh << EOF** # δημιουργία αρχείου εντολών
  cd \$1
  ls -la
  pwd
  EOF
- **$ source many.sh /etc** # χρήση του αρχείου **many.sh** σαν πηγή εντολών
- **$. many.sh /etc** # πανομοιότυπη εντολή με την πιο πάνω. Η **source** αντικαθίσταται με ”.”
- **$ bash many.sh /etc** # το αποτέλεσμα αυτής της εντολής είναι το ίδιο με πιο πάνω αλλά οι εντολές εκτελούνται σε θυγατρικό κέλυφος

Δημιουργία εκτελέσιμου αρχείου σεναρίων (scripts)

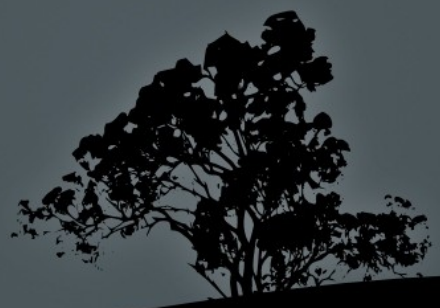
- **$ chmod a+x many.sh** # μετατροπή του προηγούμενου αρχείου σεναρίου **many.sh** σε εκτελέσιμο αρχείο
- **$ ls -l many.sh**

  -rw**x**r-**x**r-**x** 1 theo theo 17 2011-12-18 10:24 many.sh
- **$ ./many.sh** # εφόσον η εντολή **many.sh** δεν υπάρχει στο **$PATH** θα πρέπει να καλεστεί συγκεκριμένα με "**./**" ή χρησιμοποιώντας την απόλυτη τοποθεσία: **$HOME/many.sh** ή **~/many.sh**. Αν καλεστεί απλά με το όνομα του από το τρέχον κατάλογο δεν θα εκτελεστεί

Σημείωση: Η εφαρμογή των **SUID**, **SGID** σε αρχεία σεναρίων δεν έχει κανένα αποτέλεσμα. Αυτό συμβαίνει για λόγους ασφαλείας

Η γραμμή Shebang

- Το shebang είναι μια ειδική γραμμή στην πρώτη γραμμή κάθε σεναρίου που καθορίζει το πρόγραμμα που εκτελεί τις εντολές που ακολουθούν
  - **#!/bin/sh** (γενικό sh κέλυφος)
  - **#!/bin/bash** (κέλυφος bash)
  - **#!/bin/csh** (κέλυφος csh)
  - **#!/bin/tcsh** (κέλυφος tcsh)
  - **#!/bin/sed** (σενάριο sed)
  - **#!/usr/bin/awk** (σενάριο awk)
  - **#!/usr/bin/perl** (σενάριο perl)
  - **#!/usr/bin/env python** (σενάριο python)



Εφαρμογή Shebang σε αρχεία κελύφους

- **$ cat > many.sh << EOF** # δημιουργία αρχείου σεναρίου κελύφους

  #!/bin/sh

  # (Η πιο πάνω γραμμή θα μπορούσε να ήταν και **#!/bin/bash**)

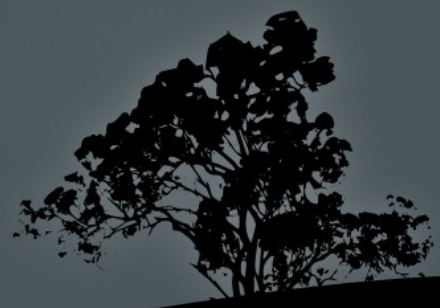
  cd \$1

  ls -la

  pwd

  EOF

- **$ chmod a+x many.sh**
- **$ ./many.sh /etc**



Αντικατάσταση Εντολών (Command Substitution)

- Για την αντικατάσταση εντολών χρησιμοποιούμε τους τελεστές ”``” ή ”$()”. Αυτές οι εντολές εκτελούνται σε θυγατρικό κέλυφος.
- **$ KERNEL_VER=`uname -r`** # το αποτέλεσμα της εντολής **uname -r** περνά σαν τιμή της μεταβλητής **KERNEL_VER**
- **$ grep -i linux $(find /usr/share/doc -name "*.txt")** # τα αποτελέσματα της εντολής **find /usr/share/doc -name "*.txt"** περνούν σαν αρχεία προς αναζήτηση στην εντολή **grep -i linux**

Αποστολή Ηλεκτρονικής Αλληλογραφίας από το κέλυφος

- **$ echo "Universe Collapse\!" | mail -s "Universe failed" root**
  # αποστολή μηνύματος στο χρήστη root με θέμα **"Universe failed"** και περιεχόμενο **"Universe Collapse\!"**
- **$ cat /var/log/messages | mail -s "Logs" user@example.com**
  # αποστολή μηνύματος στο χρήστη **user@example.com** με θέμα **"Logs"** και περιεχόμενο το αρχείο **/var/log/messages**
- **$ mail -s "File systems" user@example.com < /etc/fstab**
  # αποστολή μηνύματος στο χρήστη **user@example.com** με θέμα **"File systems"** και περιεχόμενο το αρχείο **/etc/fstab**
- **$ mail -s "Test mail" root@server.int << EOF** # άλλο παράδειγμα χρησιμοποιώντας "<<"

  > This is a test

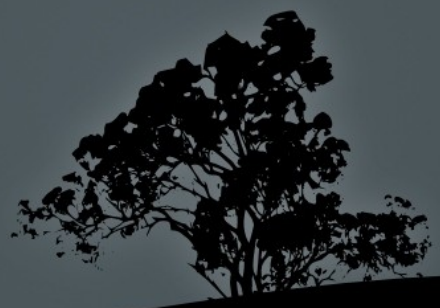
  > EOF

Εισαγωγή δεδομένων με *read*

- **$ vi user.sh** # πατήστε το "**i**" για εισαγωγή στη λειτουργία εισόδου

```
#!/bin/bash
echo "User Name: "
read USERN
echo "Shell: "
read SHELLU
echo "User Name = $USERN, Shell = SHELLU"
exit 0
```

- **$ chmod +x user.sh**
- **$ ./user.sh**



Έλεγχος κατάστασης εξόδου με *test* ή *[*

- Η εντολή **test** ή [υπάρχει σαν εσωτερική στο **bash** αλλά και σαν εκτελέσιμο αρχείο στο **$PATH**
- **$ test -e /etc/fstab** # η κατάσταση εξόδου είναι ”**0**” αν υπάρχει το αρχείο και ”**1**” αν δεν υπάρχει
- **$ [-e /etc/fstab]** # αυτή η εντολή είναι πανομοιότυπη με την πιο πάνω. Το ”[” είναι απλά ένα άλλο όνομα για την **test** με την διαφορά ότι πρέπει να τερματίζεται με ”]”. Οι αγκύλες θα πρέπει να διαχωρίζονται από το υπόλοιπο κείμενο με διαστήματα!
- **$ test -x /bin/ls** # έλεγχος αν το αρχείο είναι εκτελέσιμο
- **$ [-s ~/.bash_profile]** # έλεγχος αν υπάρχει το αρχείο και αν το μέγεθος του δεν είναι μηδέν

Έλεγχος κατάστασης εξόδου με *test* ή *[*

- **$ test "$HISTSIZE" -eq 1000** # έλεγχος αν η μεταβλητή **$HISTSIZE** ισούται με 1000. Συνίσταται οι μεταβλητές να περιβάλλονται με διπλά εισαγωγικά: ""
- **$ ["$EDITOR"]** # = **[-n "$EDITOR"]**. Έλεγχος αν η μεταβλητή **$EDITOR** έχει οριστεί
- **$ [-x /bin/ip -o -x /sbin/ip]** # λογικό OR. Έλεγχος αν υπάρχουν και είναι εκτελέσιμα τα αρχεία **/bin/ip** ή **/sbin/ip**
- **$ ["$CONT" = "yes" -a -f /usr/lib/libtest.so]** # λογικό AND. Έλεγχος αν υπάρχει η μεταβλητή **$CONT** που να ισούται με **"yes"** και να υπάρχει το κανονικό αρχείο **/usr/lib/libtest.so**

Επιλογές εντολής *test* ή *[*

- **-e file** # έλεγχος αν υπάρχει το file
- **-f file** # έλεγχος αν υπάρχει το κανονικό αρχείο file
- **-d dir** # έλεγχος αν υπάρχει ο κατάλογος dir
- **-L file** # έλεγχος αν υπάρχει ο συμβολικός σύνδεσμος file
- **-r file** # έλεγχος αν υπάρχει και αν μπορεί να διαβαστεί (readable) το αρχείο file
- **-w file** # έλεγχος αν υπάρχει και αν μπορεί να εγγραφεί (writeable) το αρχείο file
- **-x file** # έλεγχος αν υπάρχει και αν μπορεί να εκτελεστεί (executable) το αρχείο file
- **-s file** # έλεγχος αν υπάρχει και αν έχει μέγεθος μεγαλύτερο από 0 το αρχείο file
- **file1 -ot file2** # έλεγχος αν το file1 είναι παλαιότερο από το file2
- **file1 -nt file2** # έλεγχος αν το file1 είναι νεότερο από το file2

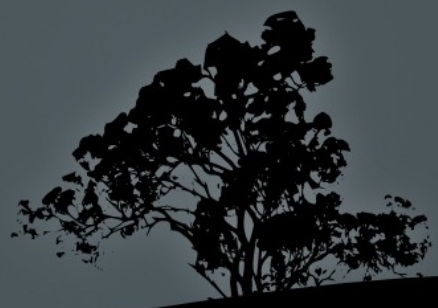
Επιλογές εντολής *test* ή *[*

- **-n string** # έλεγχος αν το μήκος της συμβολοσειράς string είναι μεγαλύτερο του 0
- **-z string** # έλεγχος αν το μήκος της συμβολοσειράς string είναι ίσο με 0
- **string1 = string2** # έλεγχος αν δύο συμβολοσειρές είναι ίδιες
- **string1 != string2** # έλεγχος αν δύο συμβολοσειρές δεν είναι ίδιες
- **arg1 -eq arg2** # το arg1 ισούται αριθμητικά με arg2
- **arg1 -ne arg2** # το arg1 είναι διαφορετικό με arg2
- **arg1 -lt arg2** # το arg1 είναι μικρότερο από arg2
- **arg1 -le arg2** # το arg1 είναι μικρότερο ή ίσο με arg2
- **arg1 -gt arg2** # το arg1 είναι μεγαλύτερο από arg2
- **arg1 -ge arg2** # το arg1 είναι μεγαλύτερο ή ίσο με arg2

Επιλογές εντολής *test* ή *[*

- **! expr** # έλεγχος αν το expr είναι εσφαλμένο (false)
- **expr1 -a expr2** # λογικό AND μεταξύ expr1 και expr2
- **expr1 -o expr2** # λογικό OR μεταξύ expr1 και expr2

Σημείωση: για περισσότερες πληροφορίες κοιτάξετε την τεκμηρίωση της **test** με:
info coreutils 'test invocation'



Εντολές υπό συνθήκη με *if*

- Το if χρησιμοποιείται για εκτέλεση εντολών υπό συνθήκη:

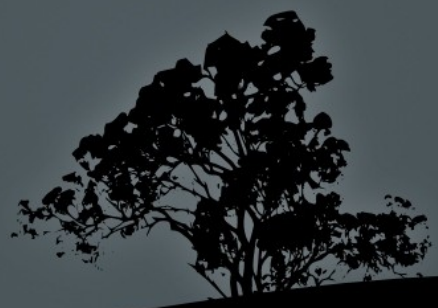
```
if [ -z "$USER" ] # = if test -z "$USER"
then
    echo \$USER is not defined!
    exit 1
elif [ "$USER" = root ]
    then
    echo 'Warning\! You are root!'
else
    echo "\$USER is $USER"
fi
```

- **$ if ["$USER" = user] ; then echo \$USER is user ; fi**

Εντολές υπό συνθήκη με *if*

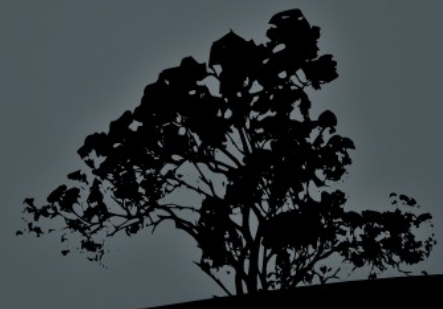
- Η **if** συνδιάζεται με οποιανδήποτε εντολή πχ **grep**. Μπορεί να καλεστεί διαδραστικά από το κέλυφος:

```
$ if grep tobedeleted /tmp/dummy.file
> then
>   rm -f /tmp/dummy.file
> elif [ "$?" = 1 ]
> then
>   echo "dummy.file is not to be deleted\!"
> else
>   echo "Error in grep"
> fi
```



Εκτύπωση αριθμητικών σειρών με *seq*

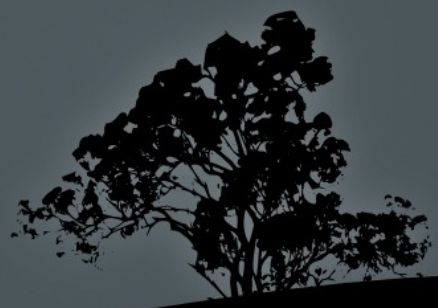
- **$ seq 1 10** # τυπώνει όλους τους αριθμούς από 1 μέχρι 10 σε ξεχωριστή γραμμή το καθένα
- **$ seq 1 2 10** # τυπώνει όλους τους αριθμούς αν;a **2** από το 1 μέχρι 10 (1, 3, 5, 7, 9)
- **$ seq 2 2 10 | xargs** # τυπώνει τους αριθμούς 2, 4, 6, 8, 10 σε μια γραμμή λόγω του **xargs**
- **$ seq 5 5 105** # πέντε, δέκα, δεκαπέντε … εκατό, εκατό πέντε



Δημιουργία βρόγχων με *for*

- Στην βασική του μορφή το **for** ορίζει μια μεταβλητή (**$PET**) που παίρνει τιμές από μια λίστα (**dog cat iguana turtle**)

```
for PET in dog cat iguana turtle
do
    echo "Pet is $PET"
done
```



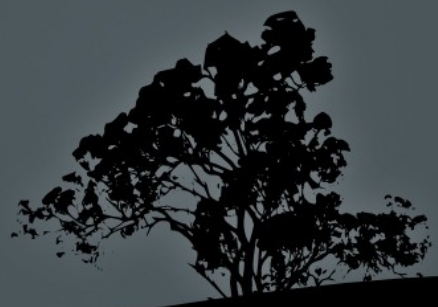
Δημιουργία βρόγχων με *for*

- **for FILE in `ls /etc`** # χρήση των περιεχομένων του **/etc** σαν λίστα

```
do
    echo "File is $FILE"
done
```

- **SUM=0**

```
for I in $(seq 1 30) # = for I in {1..30}, for ((I = 1 ; I <= 30 ; I++))
do
    SUM=`expr $I + $SUM`
    if [ "$I" -eq 30 ]
        then echo "Sum is $SUM"
    fi
done
```



Δημιουργία βρόγχων με *for*

- **$ for FILE in *** # επιλογή όλων των αρχείων/καταλόγων στον τρέχον κατάλογο

  do

  echo "$FILE is in the current directory"

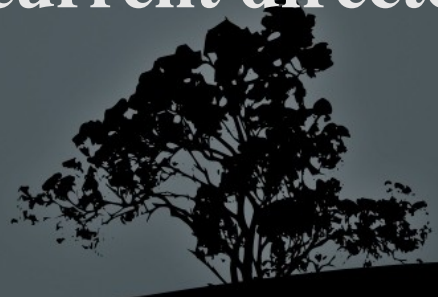
  done

- **$ for FILE in *.txt** # είναι επίσης δυνατό να χρησιμοποιήσουμε **file globbing**

  do

  echo "$FILE is a text file, in the current directory"

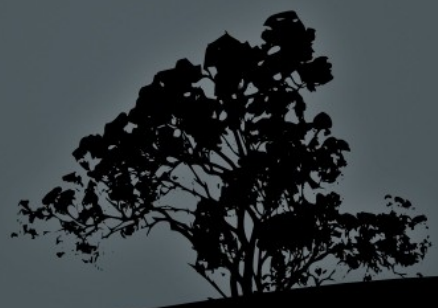
  done



Δημιουργία βρόγχων με *while*

- Το **while** χρησιμοποιείται για να ελέγξει αν μια συνθήκη στην αρχή του βρόγχου είναι αληθής και επαναλαμβάνεται ο βρόγχος μέχρι να είναι ψευδής

- ```
  VAR=0
  LIMIT=30
  while [ "$VAR" -lt "$LIMIT" ]
  do
      echo "\$VAR = $VAR"
      VAR=`expr $VAR + 1`
  done
  ```


  ```

Δημιουργία βρόγχων με *while*

- **while ["$VAR" != "end"]** # αυτός ο βρόγχος θα δέχεται τιμές και θα τις εκτυπώνει μέχρι κάποιος να εισάγει τη λέξη **"end"**

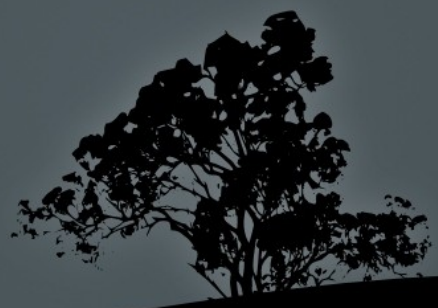
  do

  echo "Input VAR: (end to exit) "

  read VAR

  echo "\$VAR = $VAR"

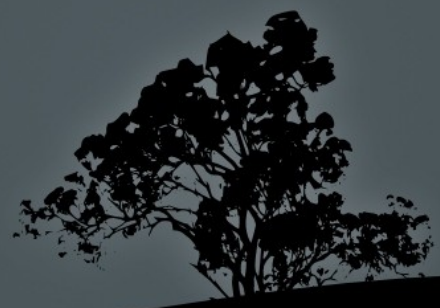
  done



Δημιουργία βρόγχων με *until*

- **Το until** χρησιμοποιείται αντίθετα με το **while** για να ελέγξει αν μια συνθήκη στην αρχή του βρόγχου είναι ψευδής και επαναλαμβάνεται ο βρόγχος μέχρι να είναι αληθής
- **until ["$VAR" = "end"]** # αυτός ο βρόγχος θα δέχεται τιμές και θα τις εκτυπώνει μέχρι κάποιος να εισάγει τη λέξη **"end"**

```
do
    echo "Input VAR: (end to exit) "
    read VAR
    echo "\$VAR = $VAR"
done
```



Εργαστήριο 2

Ξεκινήστε και τις δύο εικονικές μηχανές και συνδεθείτε σαν ”user”

- **$ cat > many.sh << EOF** # δημιουργία αρχείου εντολών
  cd \$1
  ls -la
  pwd
  EOF
- **$ source many.sh /etc**
- **$. many.sh /etc**
- **$ bash many.sh /etc**
- **$ chmod a+x many.sh**
- **$ ls -l many.sh**
- **$ many.sh /etc**
- **$ ./many.sh /etc**
- **$ ~/many.sh /usr**
- **$ $HOME/many.sh /var**
- **$ /home/user/many.sh /tmp**
- **$ vi many.sh** # και προσθέστε το shebang σαν πρωτη γραμμή.
- **$ KERNEL_VER=`uname -r`**
- **$ grep -i linux $(find /usr/share/doc \ -name "*.txt")**
- **$ echo $EDITOR**
- **$ $(export EDITOR=nano)**
- **$ echo $EDITOR**
- **$ test $EDITOR ; echo $?**
- **$ export EDITOR=vi**

Εργαστήριο 2

- $ echo $EDITOR
- $ test $EDITOR ; echo $?
- $ touch /tmp/dummy.file
- $ if grep tobedeleted /tmp/dummy.file
  then rm -f /tmp/dummy.file
  elif ["$?" = 1]
  then
  echo "dummy.file not to be \
  deleted\!";
  else echo "Error in grep\!"
  fi
- $ echo "tobedeleted" >> \
  /tmp/dummy.file
- $ if grep tobedeleted /tmp/dummy.file
  then rm -f /tmp/dummy.file
  elif ["$?" = 1]
  then
  echo "dummy.file not to be \
  deleted\!";
  else echo "Error in grep\!"
  fi
- $ ls -l /tmp/dummy.file
- $ if grep tobedeleted /tmp/dummy.file
  then rm -f /tmp/dummy.file
  elif ["$?" = 1]
  then
  echo "dummy.file not to be \
  deleted\!";
  else echo "Error in grep\!"
  fi

Εργαστήριο 2

- ```
  $ SUM=0
  $ for I in $(seq 1 30)
  do
      SUM=`expr $I + $SUM`
      if [ "$I" -eq 30 ]
      then
          echo "Sum is $SUM"
      fi
  done
  ```

- ```
  $ while [ "$VAR" != "end" ]
  do
      echo "Input VAR: (end to exit) "
      read VAR
      echo "\$VAR = $VAR"
  done
  ```