5ο ΕΡΓΑΣΤΗΡΙΟ

ΣΤΟΧΟΣ

Σενάρια φλοιού (if, while, test, error handling)

ΑΣΚΗΣΗ 1

(έχει μπει σε παλαιότερες εξετάσεις ως θέμα με παραλλαγές)

Φτιάξτε ένα πρόγραμμα σεναρίου κελύφους (script) με όνομα dirstat, το οποίο για κάθε αρχείο (δηλαδή για κάθε οντότητα του Συστήματος Αρχείων) από τον τρέχον φάκελο να ελέγχει αν είναι αρχείο, αν είναι κατάλογος ή κάτι άλλο. Πχ να εμφανίσει:

```
file1 is file
Desktop is directory
Public is something else
file2 is file
testfile is file
dirstat is file
…
```

Θυμηθείτε ότι το * (ή οποιαδήποτε έκφραση με wildcards) μεταφράζεται από το shell και αντικαθίσταται με τα ονόματα αρχείων που ταιριάζουν στο πρότυπο.

Hint: **`for i in *; do …… done`**

Η εντολή test μπορεί να χρησιμοποιηθεί να ελέγξει τον τύπο κάποιου αρχείου (-f, -d, κτλ)

Οι έλεγχοι στο shell γίνονται με συνδυασμό της εντολής if και της εντολής test.

Η εντολή test κάνει τους ελέγχους.

Σύνταξη: test έλεγχος

Εναλλακτικά της εντολής test, για λόγους "ομορφιάς" χρησιμοποιείται η εντολή [. Για λόγους συμμετρίας, αν χρησιμοποιήσουμε την εντολή [, απαιτείται το τελευταίο όρισμα να είναι το "]".

Για να επιβεβαιώσουμε ότι η [είναι εντολή, μπορούμε να βρούμε το εκτελέσιμο της:

```
asidirop@antonis-PC:~$ ls -l /usr/bin/test
-rwxr-xr-x 1 root root 26148 2011-02-23 15:22 /usr/bin/test
asidirop@antonis-PC:~$ ls -l /usr/bin/[
-rwxr-xr-x 1 root root 30244 2011-02-23 15:22 /usr/bin/[
```

Ορίσματα της test (δείτε το manual page για ολοκληρωμένη λίστα):

String comparison	
-z string	True if *string* is empty.
-n string	True if *string* is not empty.
string1 = string2	True if *string1* equals *string2*.
string1 != string2	True if *string1* does not equal *string2*.
Integer comparison	
INTEGER1 -eq INTEGER2	INTEGER1 is equal (=) to INTEGER2
INTEGER1 -ge INTEGER2	INTEGER1 is greater than or equal (>=) to INTEGER2
INTEGER1 -gt INTEGER2	INTEGER1 is greater than (>) INTEGER2
INTEGER1 -le INTEGER2	INTEGER1 is less than or equal (<=) to INTEGER2
INTEGER1 -lt INTEGER2	INTEGER1 is less than (<) INTEGER2
INTEGER1 -ne INTEGER2	INTEGER1 is not equal (!=) to INTEGER2
Filesystem Checks	
-d file	True if *file* is a directory.
-e file	True if *file* exists.
-f file	True if *file* exists and is a regular file.
-L file	True if *file* is a symbolic link.
-r file	True if *file* is a file readable by you.
-w file	True if *file* is a file writable by you.
-x file	True if *file* is a file executable by you.
-s file	true if file exists and has a size greater than zero.
file1 -nt file2	True if *file1* is newer than (according to modification time) *file2*
file1 -ot file2	True if *file1* is older than *file2*
Logical Operations	
!	unary **negation** operator.
-a	binary **and** operator.
-o	binary **or** operator (-a has higher precedence than -o).
(expr)	parentheses for grouping.

Η εντολή test κάνει τον έλεγχο που της δίνουμε σαν όρισμα, και ανάλογα επιστρέφει true (0) αν το αποτέλεσμα του ελέγχου είναι αληθές και false (!0) αν το αποτέλεσμα του ελέγχου είναι ψευδές.

Πχ αν δώσουμε τις παρακάτω εντολές:

```
asidirop@antonis-PC:~$ test 5 -gt 1
asidirop@antonis-PC:~$ echo $?
0
asidirop@antonis-PC:~$ test 5 -gt 10
asidirop@antonis-PC:~$ echo $?
```

```
1
asidirop@antonis-PC:~$ [ 5 -gt 10 ]
asidirop@antonis-PC:~$ echo $?
1
asidirop@antonis-PC:~$
```

Η μεταβλητή $? περιέχει κάθε φορά το "exit status" της τελευταίας εντολής που εκτελέστηκε (είναι μεταβλητή μιας χρήσης).

Η εντολή if δέχεται σαν "όρισμα" μια άλλη εντολή με τα ορίσματά της, και ελέγχει το exit status της εντολής. Αν η εντολή επέστρεψε 0 (δηλαδή true – επιτυχημένη εκτέλεση) τότε εκτελεί τον κώδικα στο then, διαφορετικά εκτελεί το else. Πχ:

```
asidirop@antonis-PC:~$ if ls ; then echo 'OK with files'; else echo
'Something wrong'; fi
linux-aithsh2010.pdf        linux-aithsh2011.pdf
Adopse_videos               linux-aithsh2012.pdf
aetos_data                  multipageproject
ascii.pdf                   multipageproject.crop.pdf
Audiobooks                  Music
OK with files
asidirop@antonis-PC:~$ if ls /hxxxxx; then echo 'OK with files'; else echo
'Something wrong'; fi
ls: cannot access /hxxxxx: No such file or directory
Something wrong
asidirop@antonis-PC:~$
```

Η σύνταξη της δομής if έχει ως εξής:

```
If entolh-elegxou ; then
     Commands….
elif entolh-elegxou ; then
     Commands….
elif entolh-elegxou ; then
     Commands….
else
     Commands….
fi
```

Τα τμήματα 'elif' μπορούν να είναι όσα επιθυμούμε, όπως επίσης και κανένα.

Το τμήμα 'else' μπορεί να είναι το πολύ ένα ή κανένα.

Υποχρεωτικά το block της if τερματίζει-ολοκληρώνεται με το fi. Πχ:

```
if ls "$dir" ; then
      echo "OK"
fi
if test "$a" = "5"; then
      echo "a is 5"
else
      echo "a is not 5"
fi

if [ "$b" -lt 10 ] ; then
      echo "b is small"
elif  [ "$b" -lt 100 ] ; then
      echo "b is medium"
else
      echo "b is large"
fi
```

Προσοχή στα κενά (spaces). Η εντολή if από τα υπόλοιπα χωρίζεται με ένα space τουλάχιστον. Στην εντολή ελέγχου τα ορίσματα χωρίζονται με ένα τουλάχιστον space μεταξύ τους και με ένα τουλάχιστον space από την εντολή:

```
if test "$a" = "5"; then

if [ "$b" -lt 10 ] ; then
```

σημείωση πως η "αγκύλη που κλείνει"], είναι και αυτή όρισμα στην εντολή "αγκύλη που ανοίγει" [, άρα χωρίζεται από τα προηγούμενα ορίσματα με τουλάχιστον ένα space.

Πριν και μετά το ; δεν παίζει ρόλο αν βάλουμε space ή όχι, διότι το ; είναι ειδικός χαρακτήρας (διαχωριστής εντολών).

Η εντολή **while**, δουλεύει με ακριβώς τον ίδιο τρόπο με την if, δηλαδή δέχεται ως όρισμα μια εντολή με τα ορίσματά της. Συνήθως αυτή η εντολή είναι η test.

```
while εντολή-ελέγχου ; do
      Commands…
done
πχ μια while που θέλουμε να κάνει 10 επαναλήψεις:

i=1
while [ "$i" -le 10 ] ; do
      echo $i
      i=$((i+1))
done
```

ΑΣΚΗΣΗ 2

1. Φτιάξτε ένα πρόγραμμα σεναρίου κελύφους (script) με όνομα create_test, που να δέχεται ένα όρισμα το οποίο θα εκφράζει όνομα αρχείου (παράδειγμα: ./create_test dokimi), και να δημιουργεί 50 αρχεία της μορφής file.1 file.2.
   Για το παράδειγμα αν εκτελέσουμε:
   `./create_test dokimi`

   Να μας φτιάξει τα αρχεία dokimi.1 domiki.2 …dokimi.50

   Αρχικά θα πρέπει να ελέγχει αν δόθηκε ένα ακριβώς όρισμα. Αν δεν έδωσε ο χρήστης ένα ακριβώς όρισμα να εμφανίζει μήνυμα λάθους. (Προσοχή, τα μηνύματα λαθών πρέπει να τυπώνονται στην έξοδο λαθών)

   Μπορούμε να κάνουμε την echo να τυπώσει στο stderr ανακατευθύνοντας το stream 1 στο 2. Δηλαδή:

   ```
   echo "this is an error message" 1>&2
   ```

2. Φτιάξτε ένα πρόγραμμα σεναρίου κελύφους (script) με όνομα creator100, που να δημιουργεί 100 αρχεία με την ονομασία "i"file, δηλαδή
   `1file 2file 3file … 100file`

   Αλλάξτε το script ώστε να διαγράψει τα αρχεία που δημιούργησε.

   (for ή while, string concatenation)

ΑΣΚΗΣΗ 3

Να γράψετε ένα σενάριο φλοιού (shell script) με όνομα dir_split, που :

1. θα δέχεται ως όρισμα εισόδου το όνομα ενός καταλόγου (να θεωρήσετε ότι ο κατάλογος αυτός υπάρχει και δεν απαιτείται κάποιος έλεγχος για την ύπαρξή του).
2. θα δημιουργεί δύο νέους καταλόγους με όνομα της επιλογής σας μέσα στον αρχικό κατάλογο (απαιτείται έλεγχος για το αν υπάρχουν ήδη τα ονόματα των 2 καταλόγων).
3. θα μετακινεί όλα τα αρχεία του καταλόγου στους δύο νέους καταλόγους ως εξής : Ο ένας κατάλογος θα περιλαμβάνει όλα τα αρχεία με όνομα που ξεκινά από τα γράμματα 0-5 ενώ ο άλλος κατάλογος τα υπόλοιπα (6-9). Τα αρχεία που το όνομά τους δεν ξεκινά από αριθμό, δεν θα επηρεάζονται.
4. θα εμφανίζει στην οθόνη το πλήθος των αρχείων σε καθένα από τους δύο νέους καταλόγους

ΑΣΚΗΣΗ 4

Να γραφεί ένα shell script με όνομα myfind, που θα δέχεται ως όρισμα της γραμμής εντολών ένα όνομα αρχείου. Το script θα ελέγχει αν το όνομα του αρχείου υπάρχει στον τρέχοντα κατάλογο ή σε κάποιον από τους υποκαταλόγους πρώτου επιπέδου κάτω από τον τρέχοντα κατάλογο. Το script θα εμφανίζει κάθε κατάλογο στον οποίο υπάρχει αυτό το όνομα αρχείου.

ΑΣΚΗΣΗ 5

Να γράψετε ένα σενάριο φλοιού (shell script) που :

1. θα δέχεται ως ορίσματα εισόδου :
    - το όνομα ενός καταλόγου (να ελέγξετε την ύπαρξη του καταλόγου).
    - μια επέκταση ονόματος αρχείων (πχ txt)
2. θα ελέγχει το πλήθος των ορισμάτων εισόδου
3. θα βρίσκει και θα εμφανίζει όλα τα ονόματα των αρχείων του καταλόγου που δόθηκε τα οποία:
    - έχουν την ίδια επέκταση με το δεύτερο όρισμα εισόδου και
    - τα δικαιώματα πρόσβασης περιλαμβάνουν δύο τουλάχιστον προσβάσεις r (read)
4. Το σενάριο θα εμφανίζει στο τέλος το πλήθος αυτών των αρχείων

ΑΣΚΗΣΗ 6

Να γραφτεί πρόγραμμα σεναρίου κελύφους με όνομα checkdirs. Το πρόγραμμα σεναρίου κελύφους να παίρνει ένα και μόνο όρισμα, το οποίο πρέπει να είναι όνομα καταλόγου. Να ελέγχει το πλήθος ορισμάτων και αν υπάρχει αυτός ο κατάλογος – αν δεν υπάρχει εμφανίζει κατάλληλο μήνυμα λάθους.

Αν υπάρχει ο κατάλογος να βρίσκει τα αρχεία που υπάρχουν σε αυτόν, εξαιρώντας τους υπο-καταλόγους, και με την βοήθεια της εντολής file, να μας εμφανίζει τον τύπο του κάθε αρχείου.

(Χρησιμοποιήστε την test –f και την for file in "$dir/"*; do ..... done)

ΑΣΚΗΣΗ 7

Να δημιουργηθεί μέσω editor ένα αρχείο κειμένου με όνομα data.txt σε κάθε γραμμή του οποίου θα περιλαμβάνονται 4 πεδία με διαχωριστικό : (άνω και κάτω τελεία). Τα πεδία θα είναι :

1. μια IP διεύθυνση π.χ. 195.92.211.47, 192.93.208.2 κλπ.
2. ο κωδικός του HY που έχει αυτήν την IP διεύθυνση με μήκος 5 χαρακτήρων από τους οποίους οι 3 πρώτοι χαρακτήρες δηλώνουν την αίθουσα, και οι τελευταίοι 2 τον αύξοντα αριθμό του υπολογιστή.
3. η χωρητικότητα σκληρού δίσκου που διαθέτει ο HY σε Gb π.χ. 20, 40, 100 κλπ

π.χ. ένα αρχείο μπορεί να είναι :

```
195.92.211.47:21029:400
195.92.211.48:21023:400
195.92.211.49:21022:400
195.92.211.44:21021:400
195.92.211.42:21120:500
195.92.211.7:21129:500
195.92.211.14:20121:300
195.92.211.12:20120:550
195.92.211.17:20129:200
195.92.211.2:20202:550
195.92.211.37:20201:200
```

Να γραφεί ένα σενάριο κελύφους (shell script) με όνομα comp_room που :

1. θα δέχεται ως ορίσματα της γραμμής εντολών το όνομα του αρχείου με τα δεδομένα (πχ: data.txt) και ως 2ο όρισμα το όνομα (αριθμό) της αίθουσας, πχ:
   `./comp_room data.txt 202`
2. θα ελέγχει την ύπαρξη των ορισμάτων εμφανίζοντας το κατάλληλο μήνυμα στην περίπτωση λάθους.
3. Θα ελέγχει την ύπαρξη και αναγνωσιμότητα του αρχείου δεδομένων εμφανίζοντας το κατάλληλο μήνυμα στην περίπτωση λάθους.
4. θα δημιουργεί ένα νέο αρχείο κειμένου με όνομα ipROOM.txt (όπου ROOM το όνομα της αίθουσας που δόθηκε). Το αρχείο πρέπει να περιλαμβάνει όλες τις γραμμές του αρχείου data.txt που αντιστοιχούν στην δεδομένη αίθουσα, ταξινομημένο ως προς τον αύξοντα αριθμό του PC. Πχ αν δοθεί το όρισμα 202, το αρχείο θα πρέπει να περιέχει τις γραμμές:

```
195.92.211.37:20201:200
195.92.211.2:20202:550
```

5. Να ελέγξετε αρχικά αν το αρχείο που θα δημιουργήσετε υπάρχει στον τρέχοντα κατάλογο και αν υπάρχει τότε να μετονομαστεί σε ipROOM.bak πριν ξεκινήσει η δημιουργία του νέου αρχείου.
6. θα εμφανίζει στην οθόνη, για το νέο αρχείο, το πλήθος των γραμμών του.

Η εντολή read, αν δεν διαβάσει τίποτε από την κανονική είσοδο (και διαβάσει απ' ευθείας το END_OF_FILE) επιστέφει κωδικό εξόδου false (!=0). Άρα μπορεί να χρησιμοποιηθεί μέσα σε μια while. Πχ:

```
#!/bin/bash
# file: read_test
i=1
while read a; do
     echo "line$i: $a"
     i=$((i+1))
done
```

Το παραπάνω, αν το εκτελέσουμε θα διαβάζει από την κανονική είσοδο (μέχρι το EOF) και θα εμφανίζει αυτό που διάβασε με αρίθμηση. Αν δεν ορίσαμε την κανονική είσοδο, προφανώς θα διαβάσει από το τερματικό. Αν ορίσαμε, τότε θα διαβάσει από εκεί που ορίσαμε:

```
./read_test < file1
cat file1 | ./read_test
```

Με τον ίδιο τρόπο θα μπορούσε μέσα στο script, η read αντί να διαβάζει από την κανονική είσοδο του script, να της δώσουμε να διαβάζει από αρχείο:

```
#!/bin/bash
i=1
```

```
file="file1"
while read a; do
     echo "line$i: $a"
     i=$((i+1))
done < "$file"
```

Με τον παραπάνω τρόπο, όλες οι εντολές που βρίσκονται μέσα στο block εντολών της while κληρονομούν ως κανονική είσοδο το αρχείο $file.

Γιατί στο παραπάνω δεν θα δούλευε σωστά το:

```
#!/bin/bash
i=1
file="file1"
while read a < "$file"; do
     echo "line$i: $a"
     i=$((i+1))
done
```

Η εντολή set, δέχεται διάφορα ορίσματα και τα καταχωρεί στις μεταβλητές $1,$2,…$9 όπως επίσης γεμίζει και τις #$, $*, $@.

Πχ μετά από την

```
asidirop@dellpc:~$ set hello world
asidirop@dellpc:~$ echo $1
hello
asidirop@dellpc:~$ echo $2
world
asidirop@dellpc:~$ echo $@
hello world
asidirop@dellpc:~$
```

Η εντολή set, λαμβάνει ως διαχωριστή εξ ορισμού το κενό (πολλαπλά κενά) ή αυτό που έχει οριστεί στην μεταβλητή IFS. Πχ:

```
asidirop@dellpc:~$ IFS=':'
asidirop@dellpc:~$ a='aaaaaa:bbbbbbbb:qqqqqqqqq'
asidirop@dellpc:~$ set $a
asidirop@dellpc:~$ echo $1
aaaaaa
asidirop@dellpc:~$ echo $3
qqqqqqqqq
asidirop@dellpc:~$
```

ΑΣΚΗΣΗ 8

Να γραφτεί πρόγραμμα σεναρίου κελύφους με όνομα checkusers. Το πρόγραμμα σεναρίου κελύφους να μην δέχεται κανένα όρισμα. Θα διαβάζει το αρχείο /etc/passwd. Η κάθε γραμμή του αρχείου αυτού είναι της μορφής:

```
asidirop:x:510:501:Antonis Sidiropoulos:/home/asidirop:/bin/tcsh
username:passwd:user_id:group_id:Full Name:home directory:default shell
```

και αντιστοιχεί μια γραμμή σε κάθε χρήστη. Για κάθε χρήστη, θέλουμε να εμφανίζουμε το username και το home directory.

Να χρησιμοποιήσετε συνδυασμό της while/read και την set.

Μπορείτε να ταξινομήσετε το αποτέλεσμα;

ΑΣΚΗΣΗ 9

Να γραφτεί πρόγραμμα σεναρίου κελύφους με όνομα με όνομα checkusers2. Το πρόγραμμα σεναρίου κελύφους να μην δέχεται κανένα όρισμα. Θα διαβάζει το αρχείο /etc/passwd. Η κάθε γραμμή του αρχείου αυτού είναι της μορφής:

```
asidirop:x:510:501:Antonis Sidiropoulos:/home/asidirop:/bin/tcsh
username:passwd:user_id:group_id:Full Name:home directory:default shell
```

και αντιστοιχεί μια γραμμή σε κάθε χρήστη. Για κάθε χρήστη, θέλουμε να εμφανίζουμε το username και το home directory ταξινομημένα.

Να χρησιμοποιήσετε την εντολή cut.

Η εντολή cut, είναι φίλτρο, διαβάζει την είσοδό της και τυπώνει στην έξοδο το αποτέλεσμα. Χρησιμοποιείται για να επιλέξει συγκεκριμένες "στήλες" από την είσοδο. Πχ:

ls –l | cut –c1-10,14-18

τυπώνει από την είσοδο τους χαρακτήρες $1^{ο}$ – $10^{ο}$ και $14^{ο}$-$18^{ο}$ από κάθε γραμμή.

cat /etc/group | cut –d ':' –f1,3

τυπώνει από την είσοδο τους χαρακτήρες το $1^{ο}$ και $3^{ο}$ πεδίο θεωρώντας ως διαχωριστή το ":".

ls -l | cut –d ' ' –f1,7

με βάση τον διαχωριστή ' ' (space) τύπωσε το $1^{ο}$ και $7^{ο}$ πεδίο. Όμως η έξοδος δεν είναι η αναμενόμενη. Η cut δεν αγνοεί τα πολλαπλά κενά (ή γενικώς πολλές επαναλήψεις του διαχωριστή). Άρα θα πρέπει να αφαιρεθούν οι πολλαπλές εμφανίσεις του space:

ls -l | tr –s ' '| cut –d ' ' –f1,7

το παραπάνω (εμφάνιση αδειών και ονομάτων αρχείων) δεν δουλεύει τέλεια σε όλες τις περιπτώσεις. Ποιες είναι οι προβληματικές περιπτώσεις;

ΑΣΚΗΣΗ 10

Να γραφτεί πρόγραμμα σεναρίου κελύφους με όνομα display_capital. Το πρόγραμμα σεναρίου κελύφους να παίρνει ένα και μόνο όρισμα, το οποίο πρέπει να είναι όνομα αρχείου και να εμφανίζει τα περιεχόμενα του αρχείου με κεφαλαίους χαρακτήρες.

ΑΣΚΗΣΗ 11

Να γραφτεί πρόγραμμα σεναρίου κελύφους με όνομα ls_capital. Το πρόγραμμα σεναρίου κελύφους να παίρνει ένα και μόνο όρισμα, το οποίο πρέπει να είναι όνομα καταλόγου και αφού κάνει τους απαραίτητους ελέγχους να εμφανίζει τα ονόματα αρχείων που περιέχει ο κατάλογος με κεφαλαία γράμματα.

Η εντολή uniq, είναι φίλτρο, διαβάζει την είσοδό της και το τυπώνει στην έξοδο, χωρίς όμως να τυπώνει όμοιες 2 γραμμές συνεχόμενες.

Χρησιμοποιείται για να τυπώσει τις μοναδικές γραμμές από ένα αρχείο σε συνδυασμό με την sort (ώστε όλες οι όμοιες γραμμές να είναι συνεχόμενες)

Επίσης με το όρισμα –c μετράει και τις εμφανίσεις. Πχ:

Από το αρχείο /etc/passwd μετρήστε πόσοι χρήστες χρησιμοποιούν το κάθε shell.

Απάντηση:

Από το αρχείο μας ενδιαφέρει η 7η στήλη:

```
cut –d ':' –f7 /etc/passwd
```

πρέπει να ταξινομήσουμε τις γραμμές:

```
cut –d ':' –f7 /etc/passwd | sort
```

Και μετά να εμφανίσουμε τις μοναδικές (και να τις μετρήσουμε):

```
cut –d ':' –f7 /etc/passwd | sort | uniq -c
```

ΑΣΚΗΣΗ 12

Να γραφτεί πρόγραμμα σεναρίου κελύφους με όνομα last_users. Το πρόγραμμα σεναρίου κελύφους να μην δέχεται κανένα όρισμα. Να χρησιμοποιεί την εντολή last (η εντολή last εμφανίζει ποιοι χρήστες έχουν κάνει login τον τελευταίο μήνα) με την βοήθεια της οποίας να μετράει πόσες φορές έκανε login ο κάθε χρήστης. Πχ στο users:

(έχει μπει σε παλαιότερες εξετάσεις ως θέμα με παραλλαγές)

```
asidirop@aetos:/tmp$ ./last_users
      6 asidirop
      8 bargouli
     18 ele51306
      2 ele51313
      4 ele51317
      3 ele51400
      5 ele51401
      5 ele51403
      2 ele51406
     17 ele51407…
```

ΑΣΚΗΣΗ 13

(έχει μπει σε παλαιότερες εξετάσεις ως θέμα με παραλλαγές)

Α. Να γραφτεί ένα σενάριο κελύφους με όνομα urlparse. Το script αυτό να δέχεται ακριβώς ένα όρισμα. Το όρισμα εκφράζει όνομα αρχείου το οποίο περιέχει URLs.

Πχ file1:

```
http://www.it.teithe.gr/kerveros/~diplomat/index.html
http://www.deitel.com/~books.cgi/os3e/
http://www.parinux.org/pipermail/linux/2001June/016876.html
http://cs.vt.labs.edu/~cs27_04/spring05/mcpherson/note/Ch1.pdf
http://www.cs.may.ie/~dtray/cs211/lecture1/Intro04.htm
http://www.csse.monash.edu.au/courseware/~cse 1303/parta/lecture/
http://www.amazon.com/exec/obidos/27/algorithms-on_trees.htm
http://www.amazon.com/exec/obidos
http://www.it.teithe.gr/~asidirop
http://www.it.teithe.gr/index.html
```

Το script θα πρέπει να κάνει τους απαραίτητους ελέγχους σε σχέση με τα ορίσματα, και να εμφανίζει μόνο τα hostnames (bold) από το αρχείο.

```
aetos_lib_30_$./parse_url file1
www.it.teithe.gr
www.deitel.com
www.parinux.org
```

Τελευταία Ενημέρωση: 1/6/2021 11:19 πμ

```
cs.vt.labs.edu
www.cs.may.ie
www.csse.monash.edu.au
www.amazon.com
www.amazon.com
www.it.teithe.gr
www.it.teithe.gr
```

Β. Δημιουργείστε το script url_parse2, το οποίο θα εμφανίζει τα μοναδικά hostnames και θα μετράει το πλήθος εμφανίσεων του καθενός.

```
aetos_lib_30_$./parse_url2 file1
      1 cs.vt.labs.edu
      2 www.amazon.com
      1 www.cs.may.ie
      1 www.csse.monash.edu.au
      1 www.deitel.com
      3 www.it.teithe.gr
      1 www.parinux.org
```

ΑΣΚΗΣΗ 14

(έχει μπει σε παλαιότερες εξετάσεις ως θέμα με παραλλαγές)

Φτιάξτε ένα script με όνομα iphost.sh το οποίο θα παίρνει ως όρισμα μια διεύθυνση ip και καλώντας την εντολή host θα εμφανίζει το hostname που αντιστοιχεί στο IP. Προσοχή η εντολή:

```
host 195.251.123.232
```

Εμφανίζει τα:

```
232.123.251.195.in-addr.arpa domain name pointer aetos.it.teithe.gr.
```

Από τα παραπάνω θέλουμε να κρατήσουμε μόνο το hostname (bold) που βρίσκεται στο τέλος.

Αν το IP που δίνουμε στην εντολή host δεν υπάρχει, τότε η εντολή επιστρέφει ένα μήνυμα του τύπου:

```
host 195.251.123.22
```

```
Host 22.123.251.195.in-addr.arpa not found: 3(NXDOMAIN)
```

Το script σας θα πρέπει να αναγνωρίζει αν η εντολή έδωσε “not found”. Σε αυτήν την περίπτωση θα πρέπει να τυπώνει απλά μια παύλα (-) .

Παράδειγμα χρήσης:

```
./iphost.sh 195.251.123.232
aetos.it.teithe.gr
./iphost.sh 195.222.222.222
```

Τελευταία Ενημέρωση: 1/6/2021 11:19 πμ

ΑΣΚΗΣΗ 15

(έχει μπει σε παλαιότερες εξετάσεις ως θέμα με παραλλαγές)

Δίνεται το αρχείο με όνομα 2012-03 (μπορείτε να το κατεβάσετε από την σελίδα των εργαστηρίων, folder: βοηθητικά αρχεία).

Το παραπάνω αρχείο είναι ένα αρχείο log από τον web server apache του τμήματος.

Κάθε γραμμή του αρχείου αντιστοιχεί σε μια «αίτηση» που έγινε προς τον web server.

Η κάθε γραμμή έχει την μορφή:

```
IP Client - - [ημερομηνία-ώρα] "Command Path Protocol" code size
"referer page" "client software"
```

Το Command στο παράδειγμά μας είναι GET, POST ή HEAD, και το Protocol είναι HTTP/1.X

Να φτιάξετε ένα script με όνομα webanalizer, το οποίο θα δέχεται ως όρισμα ένα αρχείο αυτής της μορφής και τα ορίσματα (προαιρετικά) –topip4, -topip6, -topurl.

- Εάν δοθεί το όρισμα –topip4, τότε το script θα βρίσκει το IP τύπου IPv4 που έχει κάνει τις περισσότερες αιτήσεις.
- Εάν δοθεί το όρισμα –topip6, τότε το script θα βρίσκει το IP τύπου IPv6 που έχει κάνει τις περισσότερες αιτήσεις.

Τα IPv4 έχουν την μορφή a.b.c.d, όπου a,b,c,d ένας αριθμός από 0 έως 255.

Τα IPv6 έχουν την μορφή: 2002:c386:421e::c386:421e όπου το κάθε στοιχείο είναι δεκαεξαδικός αριθμός.

- Εάν δοθεί το όρισμα –topurl, τότε θα βρίσκει την σελίδα (αρχείο) για την οποία έγιναν οι περισσότερες αιτήσεις.
  ```
  2002:c3fb:f34e::c3fb:f34e - - [01/Mar/2012:08:44:17 +0200] "GET
  /~asidirop/OS/current/theor/common.css HTTP/1.1" 404 525
  "http://aetos.it.teithe.gr/~asidirop/OS/current/theor/"
  "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64;
  Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30
  729; .NET CLR 3.0.30729; Media Center PC 6.0; CMDTDF; .NET4.0C)"
  ```
  Στην παραπάνω γραμμή η σελίδα είναι το τμήμα με τα έντονα γράμματα.
- Εάν δεν δοθεί κανένα από τα –topip4, -topip6, -topurl, τότε θα τα υπολογίζει όλα.
- Επίσης, να μπορούν να δοθούν τα ορίσματα με οποιαδήποτε σειρά και οποιονδήποτε συνδυασμό. Υποχρεωτικά όμως χρειάζεται το όρισμα που εκφράζει το όνομα αρχείου.

ΑΣΚΗΣΗ 16

Προσθέστε στο προηγούμενο script την δυνατότητα να χειρίζεται και τα εξής ορίσματα:

- -totalsize: Θα υπολογίζει το άθροισμα των bytes από όλες τις γραμμές (στην πάνω γραμμή είναι 525 bytes)
- -maxfile: Θα βρίσκει το αρχείο (σελίδα) με το μεγαλύτερο μέγεθος.
- -daily: θα εμφανίζει λίστα με τις ημερομηνίες, ταξινομημένες με βάση το πλήθος των αιτήσεων. Πχ για το αρχείο που δίνεται πρέπει να βγεί αποτέλεσμα:

```
  70 16/Mar/2012
  77 10/Mar/2012
  85 11/Mar/2012
  91 31/Mar/2012
  92 24/Mar/2012
  96 17/Mar/2012
 101 08/Mar/2012
 120 25/Mar/2012
 134 05/Mar/2012
 159 02/Mar/2012
 162 29/Mar/2012
 168 01/Mar/2012
 179 03/Mar/2012
 185 26/Mar/2012
 227 07/Mar/2012
 237 23/Mar/2012
 257 12/Mar/2012
 258 18/Mar/2012
 289 28/Mar/2012
 304 06/Mar/2012
 309 21/Mar/2012
 310 09/Mar/2012
 318 15/Mar/2012
 368 20/Mar/2012
 417 13/Mar/2012
 421 14/Mar/2012
 423 22/Mar/2012
 490 19/Mar/2012
 526 27/Mar/2012
 679 04/Mar/2012
```