

SCRIPT 1

Ο κώδικας του script :

```
1#!/bin/bash
2message="My username is `whoami`
3My operating system is `uname`
4My home directory is `realpath ~` and contains `ls -p ~ | grep -v / | wc -l` files and `ls -l ~
| grep "^d" | wc -l` directories.
5And here is the tree of my home directory: `ls -R ~`"
6
7if [ $# -eq 0 ]
8then
9    echo "No file supplied"
10elif [ $# -gt 1 ]
11then
12    echo "Only 1 argument is asked"
13else
14    PASSED=$1
15    if [ -d "${PASSED}" ] ; then
16        echo "$PASSED is a directory";
17    else
18        if [ -f "${PASSED}" ]; then
19            echo "You sent a message at ${PASSED}";
20            echo "$message" > $1
21        else
22            echo "${PASSED} is not valid";
23            exit 1
24        fi
25    fi
26fi
```

Αρχικά, από την γραμμή 2 μέχρι 5, περνάμε το μήνυμα που θέλουμε να στείλουμε στο αρχείο που δίνει ο χρήστης σε μια μεταβλητή message.

Το whoami δίνει το όνομα του χρήστη.

Το uname το όνομα του λειτουργικού συστήματος.

Το realpath ~ δίνει την πλήρη διαδρομή του προσωπικού καταλόγου.

Το ls -p ~ | grep -v / | wc -l δίνει πόσα συνολικά αρχεία υπάρχουν στον προσωπικό κατάλογο, αφού με το grep -v / δεν λαμβάνουμε υπόψιν τους υποκαταλόγους.

Το ls -l ~ | grep "^d" | wc -l δίνει πόσοι υποκατάλογοι υπάρχουν στον προσωπικό κατάλογο, αφού με το grep "^d" δεν λαμβάνουμε υπόψιν αρχεία.

Και το ls -R ~ δίνει το πλήρες δέντρο του προσωπικού καταλόγου.

Και μετά κάνουμε έλεγχο για να δούμε αν ο χρήστης έχει δώσει ως όρισμα κάποιο αρχείο. Στο πρώτο if κάνουμε έλεγχο για το αν έχει δώσει ο χρήστης το οποιοδήποτε όρισμα.

```
if [ $# -eq 0 ]
```

Αν δεν έχει δώσει, επιστρέφεται μήνυμα "No file supplied", όμως αν έχει δώσει παραπάνω από 1 ορίσματα επιστρέφεται μήνυμα "Only 1 argument is asked".

Αν έχει δώσει μόνο 1 όρισμα, γίνεται έλεγχος αν είναι directory και επιστέφεται μήνυμα ότι είναι directory :

```
if [ -d "${PASSED}" ] ; then
    echo "$PASSED is a directory";
```

Αν έχει δώσει σαν όρισμα κάτι άλλο όπως για παράδειγμα έναν αριθμό, επιστρέφεται μήνυμα ότι δεν είναι έγκυρο :

```
echo "${PASSED} is not valid";
exit 1
```


Και τέλος, αν το όρισμα που έδωσε ο χρήστης είναι αρχείο, τότε το μήνυμα που συντάξαμε θα σταλεί στο περιεχόμενο του αρχείου.

```
if [ -f "${PASSED}" ]; then
    echo "You sent a message at ${PASSED}";
    echo "$message" > $1
```

Αν τρέξουμε την εντολή :

```
q@ubuntu:~/Desktop/scripts$ bash identity /home/q/test.txt
You sent a message at /home/q/test.txt
```

Θα δούμε ότι το περιεχόμενο του test1.txt περιέχει το μήνυμα που συντάξαμε :



```
Identity × test.txt ×
1 My username is q
2 My operating system is Linux
3 My home directory is /home/q and contains 4 files and 9 directories.
4 And here is the tree of my home directory: /home/q:
5 3.txt
6 Desktop
7 Documents
8 Downloads
9 Music
10 Pictures
11 Public
12 Templates
13 test1.txt
14 testing
15 test.txt
16 text.txt
17 Videos
18
19 /home/q/Desktop:
20 scripts
21 Unix
22
23 /home/q/Desktop/scripts:
24 $
25 identity
26
27 /home/q/Desktop/Unix:
28 Egkatastasi Ubuntu Linux VMWare Workstation Player .pdf
29 Fylladio #Ασκηση με Λύση.pdf
```

SCRIPT 2

Ο κώδικας του script :

```

1#!/bin/bash
2sleep 2s
3if [ $# -eq 0 ]
4then
5    echo "No directory supplied"
6elif [ $# -gt 1 ]
7then
8    echo "Only 1 argument is asked"
9else
10    PASSED=$1
11    if [ -d "${PASSED}" ] ; then
12        count1=`find ${PASSED} -maxdepth 1 -perm /u+rw -not -path '*/\.*' -type f | wc -l`
13        echo "There are $count1 files with read and write permissions for owner";
14        count2=`find ${PASSED} -maxdepth 1 -perm /g+rw -not -path '*/\.*' -type f | wc -l`
15        echo "There are $count2 files with read and write permissions for group";
16        count3=`find ${PASSED} -maxdepth 1 ! -perm /o+rw -not -path '*/\.*' -type f | wc -l`
17        echo "There are $count3 files with no permissions for others";
18    else
19        if [ -f "${PASSED}" ]; then
20            echo "$PASSED is a file";
21        else
22            echo "${PASSED} is not valid";
23            exit 1
24        fi
25    fi
26fi

```

Στην γραμμή 2 γίνεται παύση του script για 2 δευτερόλεπτα.

Και μετά κάνουμε έλεγχο για να δούμε αν ο χρήστης έχει δώσει ως όρισμα κάποιον κατάλογο. Στο πρώτο if κάνουμε έλεγχο για το αν έχει δώσει ο χρήστης το οποιοδήποτε όρισμα.

```
if [ $# -eq 0 ]
```

Αν δεν έχει δώσει, επιστρέφεται μήνυμα "No directory supplied", όμως αν έχει δώσει παραπάνω από 1 ορίσματα επιστρέφεται μήνυμα "Only 1 argument is asked".

Αν έχει δώσει μόνο 1 όρισμα, γίνεται έλεγχος αν είναι file και επιστρέφεται μήνυμα ότι είναι file :

```
if [ -f "${PASSED}" ]; then
    echo "$PASSED is a file";
```

Αν έχει δώσει σαν όρισμα κάτι άλλο όπως για παράδειγμα έναν αριθμό, επιστρέφεται μήνυμα ότι δεν είναι έγκυρο :

```
echo "${PASSED} is not valid";
exit 1
```

Και τέλος, αν το όρισμα που έδωσε ο χρήστης είναι κατάλογος, τότε μετράμε αρχεία με τα αντίστοιχα δικαιώματα που θέλουμε :

```

if [ -d "${PASSED}" ] ; then
    count1=`find ${PASSED} -maxdepth 1 -perm /u+rw -not -path '*/\.*' -type f | wc -l`
    echo "There are $count1 files with read and write permissions for owner";
    count2=`find ${PASSED} -maxdepth 1 -perm /g+rw -not -path '*/\.*' -type f | wc -l`
    echo "There are $count2 files with read and write permissions for group";
    count3=`find ${PASSED} -maxdepth 1 ! -perm /o+rw -not -path '*/\.*' -type f | wc -l`
    echo "There are $count3 files with no permissions for others";

```

Με το -maxdepth 1, ψάχνουμε μόνο μέσα στον κατάλογο που έχει δώσει ο χρήστης.

Με το -not -path '*/\.*', αποκλείουμε κρυμμένα αρχεία από την αναζήτηση.

Με το -type f, ψάχνουμε μόνο για αρχεία.

Και με το | wc -l, μετράμε τα αρχεία.

Στο count1 έχει δοθεί -perm /u+rw, το οποίο σημαίνει ότι θέλουμε να βρούμε αρχεία που έχουν ενεργοποιημένες τις εξουσιοδοτήσεις διαβάσματος και γραψίματος για τον ιδιοκτήτη.

Στο count2 έχει δοθεί -perm /g+rw, το οποίο σημαίνει ότι θέλουμε να βρούμε αρχεία που έχουν ενεργοποιημένες τις εξουσιοδοτήσεις διαβάσματος και γραψίματος για την ομάδα.

Στο count3 έχει δοθεί ! -perm /o+rw, το οποίο σημαίνει ότι θέλουμε να βρούμε αρχεία που δεν έχουν ενεργοποιημένη καμιά εξουσιοδότηση για τους υπόλοιπους.

Για τον έλεγχο της σωστής λειτουργίας του script, στον προσωπικό κατάλογο υπάρχουν 5 αρχεία και εκτελέστηκαν οι εξής εντολές :

```
chmod 601 3.txt chmod
```

```
061 test1.txt chmod
```

```
070 test2.txt chmod
```

```
660 test.txt
```

```
chmod 664 text.txt
```

```
q@ubuntu:~$ ls -l
total 44
-rw-----x 1 q q    0 Jun 12 14:24 3.txt
drwxr-xr-x 4 q q 4096 Jun 12 12:06 Desktop
drwxr-xr-x 2 q q 4096 Jun 12 05:26 Documents
drwxr-xr-x 2 q q 4096 Jun 12 05:26 Downloads
drwxr-xr-x 2 q q 4096 Jun 12 05:26 Music
drwxr-xr-x 2 q q 4096 Jun 12 05:26 Pictures
drwxr-xr-x 2 q q 4096 Jun 12 05:26 Public
drwxr-xr-x 2 q q 4096 Jun 12 05:26 Templates
----rwx---x 1 q q 2404 Jun 13 03:06 test1.txt
----rwx--- 1 q q    0 Jun 13 04:36 test2.txt
drwxrwxr-x 2 q q 4096 Jun 12 14:24 testing
-rw-rw---- 1 q q 2404 Jun 13 03:12 test.txt
-rw-rw-r-- 1 q q    0 Jun 12 14:19 text.txt
drwxr-xr-x 2 q q 4096 Jun 12 05:26 Videos
```

Και αν τρέξουμε την εντολή θα δούμε ως αποτέλεσμα :

```
q@ubuntu:~/Desktop/scripts$ bash mygrep ~
There are 3 files with read and write permissions for owner
There are 4 files with read and write permissions for group
There are 2 files with no permissions for others
```


Που σημαίνει ότι το script δουλεύει σωστά.

SCRIPT 3

Ο κώδικας του script :

```
1 #!/bin/bash
2 totalTime=0
3 totalAmount=5000
4 totalCustomers=1
5 while [ 1 ]
6 do
7     read -p "Give time of customer (in minutes):" time
8     if ! [[ $time =~ ^[0-9]+$ ]] ; then
9         echo "Error: try again" >&2; continue
10    fi
11    read -p "Give amount of payment of customer:" amount
12    if ! [[ $amount =~ ^[-+]?[0-9]+$ ]] ; then
13        echo "Error: try again" >&2; continue
14    fi
15    totalAmount=`expr $totalAmount + $amount`
16    totalTime=`expr $totalTime + $time`
17    if [[ $totalAmount -lt 0 ]]; then
18        echo "We can't afford that amount : ${totalAmount#-} €" >&2; break
19    fi
20    if [[ $totalTime -gt 300 ]]; then
21        echo "5 hours have passed" >&2; break
22    fi
23    totalCustomers=`expr $totalCustomers + 1`
24 done
25
26 echo "Tameio has closed"
27 if [[ $totalCustomers -eq 1 ]]; then
28     echo "1 customer passed"
29 else
30     echo "${totalCustomers} total customers passed"
31 fi
32 echo "Tameio worked ${totalTime} minutes"
```

Στην αρχή αναθέτουμε τις βασικές μεταβλητές του script, που είναι η συνολική ώρα που είναι ανοιχτό το ταμείο (totalTime), το συνολικό ποσό του ταμείου (totalAmount) και οι συνολικοί πελάτες (totalCustomers).

Μετά, μπαίνουμε σε μια επαναληπτική διαδικασία την while που θα είναι true για πάντα μέχρι να γίνει break.

Ζητάμε από τον χρήστη δυο τιμές, τον χρόνο του πελάτη σε και το ποσό που θα πληρώσει ή θα πληρωθεί. Γίνεται έλεγχος αν οι τιμές είναι αριθμοί και όχι γράμμα :

```
if ! [[ $time =~ ^[0-9]+$ ]] ; then
    echo "Error: try again" >&2; continue
fi
```

Ενώ, όσον αφορά το ποσό που θα πληρώσει ή θα πληρωθεί ο πελάτης, δεχόμαστε και το – μπροστά στον αριθμό που σημαίνει ότι ο πελάτης θα πληρωθεί :

```
if ! [[ $amount =~ ^[-+]?[0-9]+$ ]] ; then
    echo "Error: try again" >&2; continue
fi
```

Μετά, προσθέτουμε τις τιμές που έδωσε ο χρήστης στις αντίστοιχες μεταβλητές που αναθέσαμε στην αρχή και γίνεται έλεγχος αν το ταμείο έχει αδειάσει :

```
if [[ $totalAmount -lt 0 ]]; then
    echo "We can't afford that amount : ${totalAmount#-} €" >&2; break
fi
```

(Το #- στο τέλος της μεταβλητής προστέθηκε για να φαίνεται η απόλυτη τιμή του αριθμού γιατί είναι αρνητικός)

Οπότε αν έχει αδειάσει φαίνεται σε μήνυμα το ποσό που αδυνατεί να πληρώσει το ταμείο και βγαίνουμε από την επαναληπτική διαδικασία while με το break.

Και γίνεται επίσης έλεγχος για το αν έχουν περάσει 5 ώρες που δούλεψε το ταμείο :

```
if [[ $totalTime -gt 300 ]]; then
    echo "5 hours have passed" >&2; break
fi
```

Οπότε αν έχουν περάσει 5 ώρες που δούλεψε το ταμείο, εμφανίζεται το αντίστοιχο μήνυμα και βγαίνουμε από την επαναληπτική διαδικασία while με το break.

Για κάθε πελάτη γίνονται και οι δυο παραπάνω έλεγχοι, αν ισχύει έστω και ένας η διαδικασία σταματάει. Όμως, αν δεν ισχύουν, δηλαδή αν δεν έχουν περάσει ακόμα 5 ώρες και το ταμείο δεν έχει αδειάσει, η διαδικασία συνεχίζει με τον επόμενο πελάτη.

Στο τέλος, όταν η διαδικασία σταματάει τυπώνεται ο αριθμός των πελατών που πέρασαν καθώς και για πόσο χρόνο δούλεψε το ταμείο :

```
echo "Tameio has closed"
if [[ $totalCustomers -eq 1 ]]; then
    echo "1 customer passed"
else
    echo "${totalCustomers} total customers passed"
fi
echo "Tameio worked ${totalTime} minutes"
```

SCRIPT 4

Στην αρχή του script γίνεται ένας έλεγχος για το πόσα ορίσματα δίνει ο χρήστης. Ειδικότερα, αν ο χρήστης δεν δώσει κανένα όρισμα τότε θα του εμφανιστεί μήνυμα "No arguments supplied", αν δώσει μόνο 1 όρισμα τότε θα του εμφανιστεί μήνυμα "2 arguments are asked" και αν δώσει παραπάνω από 2 ορίσματα τότε θα του εμφανιστεί μήνυμα "Only 2 arguments are asked" :

```

1 #!/bin/bash
2 if [ $# -eq 0 ]
3 then
4     echo "No arguments supplied"
5 elif [ $# -eq 1 ]
6 then
7     echo "2 argumenst are asked"
8 elif [ $# -gt 2 ]
9 then
10    echo "Only 2 arguments are asked"
11 else

```

Οπότε αν ο χρήστης δώσει 2 ακριβώς ορίσματα γίνεται έλεγχος για το αν τα ορίσματα είναι αριθμοί και επιπλέον για το πρώτο όρισμα, γίνεται έλεγχος αν ανήκει στο οκταδικό σύστημα :

```

11 else
12     number1=$1
13     number2=$2
14     if ! [[ $number1 =~ ^[0-9]+$ ]] ; then
15         echo "First argument must be a number"
16         exit 1
17     fi
18     if ! [[ $number2 =~ ^[0-9]+$ ]] ; then
19         echo "Second argument must be a number"
20         exit 1
21     fi
22     if ! [[ $number1 =~ ^[0-7][0-7][0-7]+$ ]] ; then
23         echo "First argument must use 3 digits 0 to 7"
24         exit 1
25     fi
26     if [ $number1 -gt 777 ] ; then
27         echo "First argument must not be above 777";
28         exit 1
29     fi
30

```

Αν δεν ισχύουν οι παραπάνω περιορισμοί, τότε το script τερματίζει και ο χρήστης θα πρέπει να το εκτελέσει ξανά.

Στην συνέχεια, γίνεται η αρχικοποίηση κάποιων μεταβλητών που θα μας βοηθήσουν να εμφανίσουμε στο τέλος του script πόσα αρχεία ή υποκαταλόγους βρέθηκαν για κάθε περίπτωση και για κάθε κατάλογο.

```

31     exitValue=0
32     totalCount1=0
33     totalCount2=0
34     totalCount3=0
35     totalCount4=0
36     totalCount5=0
37
38     declare -A arr
39     i=0

```

Στην συνέχεια :

```

41     while [ $exitValue = 0 ]
42     do
43         while [ 1 ]
44         do
45             read -p "Give directory:" directory
46             if [ -d "${directory}" ] ; then
47                 break
48             else
49                 if [ -f "${directory}" ]; then
50                     echo "${directory} is a file";
51                     continue
52                 else
53                     echo "${directory} is not valid";
54                     continue
55                 fi
56             fi
57         done
58     done

```

Το πρώτο while επιτρέπει στον χρήστη να εκτελέσει το script όσες φορές επιθυμεί (για διαφορετικούς καταλόγους), αφού στο τέλος θα ερωτάται αν θέλει να σταματήσει ή να ψάξει και άλλον κατάλογο.

Το δεύτερο while θα εκτελείται συνέχεια μέχρι ο χρήστης να δώσει κάποιον κατάλογο. Αν δώσει αρχείο ή οτιδήποτε άλλο θα του εμφανιστεί το αντίστοιχο μήνυμα λάθους και θα του ζητηθεί ξανά να δώσει κάποιον κατάλογο, ενώ αν δώσει κάποιον κατάλογο τότε θα μπορεί να προχωρήσει.

Στην συνέχεια, γίνεται αναζήτηση αρχείων ή υποκαταλόγων για τον κατάλογο που έδωσε για κάθε περίπτωση ξεχωριστά.

Γίνεται η καταμέτρηση των αρχείων για να τυπώνεται η κατάλληλη επικεφαλίδα η οποία να αναφέρει τον αριθμό των αρχείων (ή υποκαταλόγων) που πρόκειται να τυπωθούν και μετά εκτυπώνεται η λίστα με όλα τα αρχεία (ή όλους τους υποκαταλόγους).

Ειδικότερα, στην πρώτη περίπτωση γίνεται αναζήτηση των αρχείων του δέντρου του δοθέντος καταλόγου με εξουσιοδοτήσεις το πρώτο όρισμα που έδωσε ο χρήστης :


```

59 count1=`find ${directory} -perm $number1 -not -path '*/\.*' -type f | wc -l`
60 totalCount1=`expr $totalCount1 + $count1`
61 if [ $count1 -eq 1 ]; then
62     echo "1)There is $count1 file with permissions $number1";
63 else
64     echo "1)There are $count1 files with permissions $number1";
65 fi
66 find ${directory} -perm $number1 -not -path '*/\.*' -type f
67

```

Με το -not -path '*/\.*' στην αναζήτηση αποκλείονται τα κρυμμένα αρχεία.

Στην συνέχεια, γίνεται αναζήτηση των αρχείων του δέντρου του δοθέντος καταλόγου που άλλαξαν περιεχόμενα κατά τις 'x' τελευταίες μέρες, όπου 'x' το δεύτερο όρισμα που έδωσε ο χρήστης :

```

68 count2=`find ${directory} -mtime -$number2 -not -path '*/\.*' -type f | wc -l`
69 totalCount2=`expr $totalCount2 + $count2`
70 if [ $count2 -eq 1 ]; then
71     echo "2)There is $count2 file that was modified in the last $number2 days";
72 else
73     echo "2)There are $count2 files that were modified in the last $number2 days";
74 fi
75 find ${directory} -mtime -$number2 -not -path '*/\.*' -type f
76

```

Στην επόμενη περίπτωση, γίνεται αναζήτηση των υποκαταλόγων του δέντρου του δοθέντος καταλόγου που προσπελάστηκαν κατά τις 'x' τελευταίες μέρες, όπου 'x' το δεύτερο όρισμα που έδωσε ο χρήστης :

```

77 count3=`find ${directory} -mtime -$number2 -not -path '*/\.*' -type d | wc -l`
78 totalCount3=`expr $totalCount3 + $count3`
79 if [ $count3 -eq 1 ]; then
80     echo "3)There is $count3 subdirectory that was modified in the last $number2 days";
81 else
82     echo "3)There are $count3 subdirectories that were modified in the last $number2 days";
83 fi
84 find ${directory} -mtime -$number2 -not -path '*/\.*' -type d
85

```

Στην συνέχεια γίνεται αναζήτηση των αρχείων του δέντρου του δοθέντος καταλόγου που είναι τύπου pipe ή socket :

```

86 count4=`find ${directory} -not -path '*/\.*' -type p -o -type s | wc -l`
87 totalCount4=`expr $totalCount4 + $count4`
88 if [ $count4 -eq 1 ]; then
89     echo "4)There is $count4 file that is type of pipe or socket";
90 else
91     echo "4)There are $count4 files that is type of pipe or socket";
92 fi
93 find ${directory} -not -path '*/\.*' -type p -o -type s

```

Και στην τελευταία περίπτωση, γίνεται αναζήτηση των κενών αρχείων του δοθέντος καταλόγου (όχι του δέντρου) :

```

95 count5=`find ${directory} -maxdepth 1 -not -path '*/\.*' -type f -size 0 | wc -l`
96 totalCount5=`expr $totalCount5 + $count5`
97 if [ $count5 -eq 1 ]; then
98     echo "5)There is $count5 empty file";
99 else
100     echo "5)There are $count5 empty files";
101 fi
102 find ${directory} -maxdepth 1 -not -path '*/\.*' -type f -size 0
103

```

Στην συνέχεια περνάμε σε πίνακα πόσα αρχεία/υποκαταλόγους κάθε περίπτωσης βρέθηκαν για τον κατάλογο που έγινε η αναζήτηση :

```

104         arr[$i,0]=$directory
105         arr[$i,1]=$count1
106         arr[$i,2]=$count2
107         arr[$i,3]=$count3
108         arr[$i,4]=$count4
109         arr[$i,5]=$count5
110
111         i=`expr $i + 1`
112
113         echo "If you want to quit press any number except 0,if you want to look for another directory, press 0"
114         read exitValue
115     done

```

Και ο χρήστης ερωτάται αν θέλει να πραγματοποιήσει αναζήτηση και για άλλον κατάλογο ή αν θέλει να σταματήσει.

Αν σταματήσει, του εμφανίζεται αθροιστικά το συνολικό αριθμό των ευρεθέντων αρχείων (ή υποκαταλόγων) κάθε περίπτωσης (από τις 1 έως 5) για όλους τους καταλόγους στους οποίους έψαξε :

```

116     echo "Final results : "
117     if [ $totalCount1 -eq 1 ]; then
118         echo "1)You found $totalCount1 file with permissions $number1";
119     else
120         echo "1)You found $totalCount1 files with permissions $number1";
121     fi
122     if [ $totalCount2 -eq 1 ]; then
123         echo "2)You found $totalCount2 file that was modified in the last $number2 days";
124     else
125         echo "2)You found $totalCount2 files that were modified in the last $number2 days";
126     fi
127     if [ $totalCount3 -eq 1 ]; then
128         echo "3)You found $totalCount3 subdirectory that was modified in the last $number2 days";
129     else
130         echo "3)You found $totalCount3 subdirectories that were modified in the last $number2 days";
131     fi
132     if [ $totalCount4 -eq 1 ]; then
133         echo "4)You found $totalCount4 file that is type of pipe or socket";
134     else
135         echo "4)You found $totalCount4 files that is type of pipe or socket";
136     fi
137     if [ $totalCount5 -eq 1 ]; then
138         echo "5)You found $totalCount5 empty file";
139     else
140         echo "5)You found $totalCount5 empty files";
141     fi
142

```

Και του εμφανίζεται επίσης αναλυτικό ιστορικό/ανακεφαλαίωση (πόσα αρχεία/υποκαταλόγους κάθε περίπτωσης βρήκε) για κάθε κατάλογο στον οποίο έψαξε :

```

143     j=0
144
145     while [ 1 ]
146     do
147         echo "Directory : ${arr[$j,0]}"
148         if [ ${arr[$j,1]} -eq 1 ]; then
149             echo "1)You found ${arr[$j,1]} file with permissions $number1";
150         else
151             echo "1)You found ${arr[$j,1]} files with permissions $number1";
152         fi
153         if [ ${arr[$j,2]} -eq 1 ]; then
154             echo "2)You found ${arr[$j,2]} file that was modified in the last $number2 days";
155         else
156             echo "2)You found ${arr[$j,2]} files that were modified in the last $number2 days";
157         fi
158         if [ ${arr[$j,3]} -eq 1 ]; then
159             echo "3)You found ${arr[$j,3]} subdirectory that was modified in the last $number2 days";
160         else
161             echo "3)You found ${arr[$j,3]} subdirectories that were modified in the last $number2 days";
162         fi
163         if [ ${arr[$j,4]} -eq 1 ]; then
164             echo "4)You found ${arr[$j,4]} file that is type of pipe or socket";
165         else
166             echo "4)You found ${arr[$j,4]} files that is type of pipe or socket";
167         fi
168         if [ $totalCount5 -eq 1 ]; then
169             echo "5)You found ${arr[$j,5]} empty file";
170         else
171             echo "5)You found ${arr[$j,5]} empty files";
172         fi
173         j=`expr $j + 1`
174         if [ $j -eq $i ]; then
175             break
176         fi
177     done
178 fi

```

Με το while εκτελείται ο παραπάνω κώδικας για όσους καταλόγους έψαξε ο χρήστης.