

**ΕΡΓΑΣΤΗΡΙΟ 11****Βασικές εντολές UNIX**

1. Βρείτε τον αριθμό των ενεργών χρηστών του συστήματος που δουλεύετε.

`who / wc -l`
ή
`users / wc -w`

2. Πόσα αρχεία και κατάλογοι βρίσκονται στον τρέχοντα κατάλόγό σας;

`ls -l / wc -l`

Ο αριθμός που επιστρέφεται είναι κατά ένα μεγαλύτερος από τον σωστό. Η πρώτη γραμμή που επιστρέφει το `ls -l` δεν αναφέρεται σε κάποιο αρχείο ή κατάλογο και δεν πρέπει να προσμετρηθεί. Η γραμμή αυτή δείχνει το συνολικό μέγεθος των αρχείων και των καταλόγων του τρέχοντος καταλόγου σε μπλοκς (total blocks). Το μέγεθος του κάθε μπλοκ είναι 1KB.

3. Πόσα .c αρχεία βρίσκονται στον τρέχοντα κατάλόγο σας;

`ls -l *.c / wc -l`

4. Τυπώστε ταξινομημένα τα περιεχόμενα του τρέχοντος καταλόγου βάση του χρόνου τροποποίησης ξεκινώντας από το παλαιότερο, δίνοντας όλες τις πληροφορίες για κάθε καταχώρηση.

`ls -ltr`

5. Τοποθετήστε στο αρχείο `last_updated` τα 3 αρχεία ή καταλόγους του τρέχοντος καταλόγου τα οποία έχουν τύχει αλλαγών παλαιότερα.

`ls -rt / head -3 > last_updated`

6. Τυπώστε σε αντίστροφη αλφαβητική σειρά τους ενεργούς χρήστες της μηχανής σας.

`who / sort -r`

7. Εκτυπώστε στην οθόνη τις γραμμές 20-80 του αρχείου `file.txt`. Πως αλλάζει η εντολή εάν θέλουμε μπροστά από κάθε γραμμή να φαίνεται ο αριθμός της; Αν θέλουμε να εμφανίζεται σελίδα-σελίδα στην οθόνη μας;

`cat file.txt / head -80 / tail -60`
`cat -n file.txt / head -80 / tail -60`
`cat file.txt / head -80 / tail -60 / more`

Επεξήγηση:

Με τη χρήση της εντολής **cat** τυπώνουμε το αρχείο `file.txt` στο `standard output` και στη συνέχεια με την εντολή **head** παίρνουμε τις πρώτες 80 γραμμές εκ των οποίων φιλτράρουμε τις τελευταίες 60 με τη χρήση της εντολής **tail**.

- **cat** : Τυπώνουμε το αρχείο `file.txt` στο `standard output`
- **head -80** : Παίρνουμε τις πρώτες 80 γραμμές
- **tail -60** : Φιλτράρουμε τις τελευταίες 60

8. Ταξινόμησε το αρχείο `/etc/passwd`, τοποθετώντας το αποτέλεσμα στο αρχείο `foo` και οποιοδήποτε λάθος (error) στο αρχείο `err`. Αλλάξτε την εντολή σας έτσι ώστε τόσο το αποτέλεσμα όσο και τα λάθη να κατευθύνονται στο αρχείο `foo3`.

```
sort < /etc/passwd > foo 2> err ή
sort /etc/passwd > foo 2> err ή
cat /etc/passwd | sort > foo 2> err

cat /etc/passwd | sort &> foo3 ή
cat /etc/passwd | sort 1> foo3 2>>foo3 ή
cat /etc/passwd | sort > foo3 2>&1
```

The `/etc/passwd` contains one entry per line for each user (or user account) of the system. All fields are separated by a colon (:) symbol. Total seven fields as follows.

Generally, `passwd` file entry looks as follows (click to enlarge image):

oracle:x:1021:1020:Oracle user:/data/network/oracle:/bin/bash

1 2 3 4 5 6 7

(Fig.01: `/etc/passwd` file format - click to enlarge)

1. **Username:** It is used when user logs in. It should be between 1 and 32 characters in length.
2. **Password:** An x character indicates that encrypted password is stored in `/etc/shadow` file.
3. **User ID (UID):** Each user must be assigned a user ID (UID). UID 0 (zero) is reserved for root and UIDs 1-99 are reserved for other predefined accounts. Further UID 100-999 are reserved by system for administrative and system accounts/groups.
4. **Group ID (GID):** The primary group ID (stored in `/etc/group` file)
5. **User ID Info:** The comment field. It allow you to add extra information about the users such as user's full name, phone number etc. This field use by `finger` command.
6. **Home directory:** The absolute path to the directory the user will be in when they log in. If this directory does not exists then users directory becomes `/`
7. **Command/shell:** The absolute path of a command or shell (`/bin/bash`). Typically, this is a shell. Please note that it does not have to be a shell.



Your encrypted password is not stored in `/etc/passwd` file. It is stored in `/etc/shadow` file. In the good old days there was no great problem with this general read permission. Everybody could read the encrypted passwords, but the hardware was too slow to crack a well-chosen password, and moreover, the basic assumption used to be that of a friendly user-community.

9. Έχουμε τις δύο πιο κάτω εντολές

```
#1: ls | head -1  
#2: ls > temp; head -1 < temp
```

Απαντήστε στις πιο κάτω ερωτήσεις:

- 1) Γιατί η πρώτη εντολή δουλεύει στο φάκελο `/etc` ενώ η δεύτερη όχι;
- 2) Κάτω από ποιες προϋποθέσεις θα μπορούσε η δεύτερη εντολή να μην παράγει το ίδιο αποτέλεσμα με την πρώτη, όταν αυτή τρέχει σε κάποιο φάκελο του προσωπικού μας χώρου;

Η δεύτερη εντολή δημιουργεί ένα αρχείο για να κρατήσει το αποτέλεσμα της εντολής `ls` ενώ η πρώτη κάνει `pipe` στην εντολή `head`. Η δεύτερη εντολή αποτυγχάνει στην περίπτωση που στο φάκελο τον οποίο δουλεύουμε δεν έχουμε άδεια για να γράψουμε (όπως στο `etc`).

*Το αποτέλεσμα μπορεί να διαφέρει σε περίπτωση που το αρχείο `temp` είναι το πρώτο αρχείο του φακέλου, δηλαδή αν ο φάκελος δεν είχε άλλα αρχεία μέσα (το `temp` δημιουργείται πριν να τρέξει η εντολή `ls`). Επίσης στην περίπτωση που στο φάκελο που βρισκόμαστε υπάρχει αρχείο με το όνομα `temp` και η μεταβλητή `noclobber` έχει γίνει `set`, τότε το αρχείο `temp` δεν θα επικαλυφτεί (*overwritten*) με αποτέλεσμα οι εντολές να επιστρέψουν την πρώτη γραμμή του αρχείου `temp`.*

Σε `bash shell`:

```
ls > test
```

```
set -o noclobber
```

```
ls > test
```

```
bash: test: cannot overwrite existing file
```

```
set +o noclobber
```

```
ls > test
```