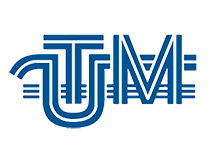
****

*Technical University of Moldova*

Cybersecurity lab 1

**Submitted by**:Emil Terman *FAF 161*

**Professor**: Cazac M.

**2019**

**Ex1.**

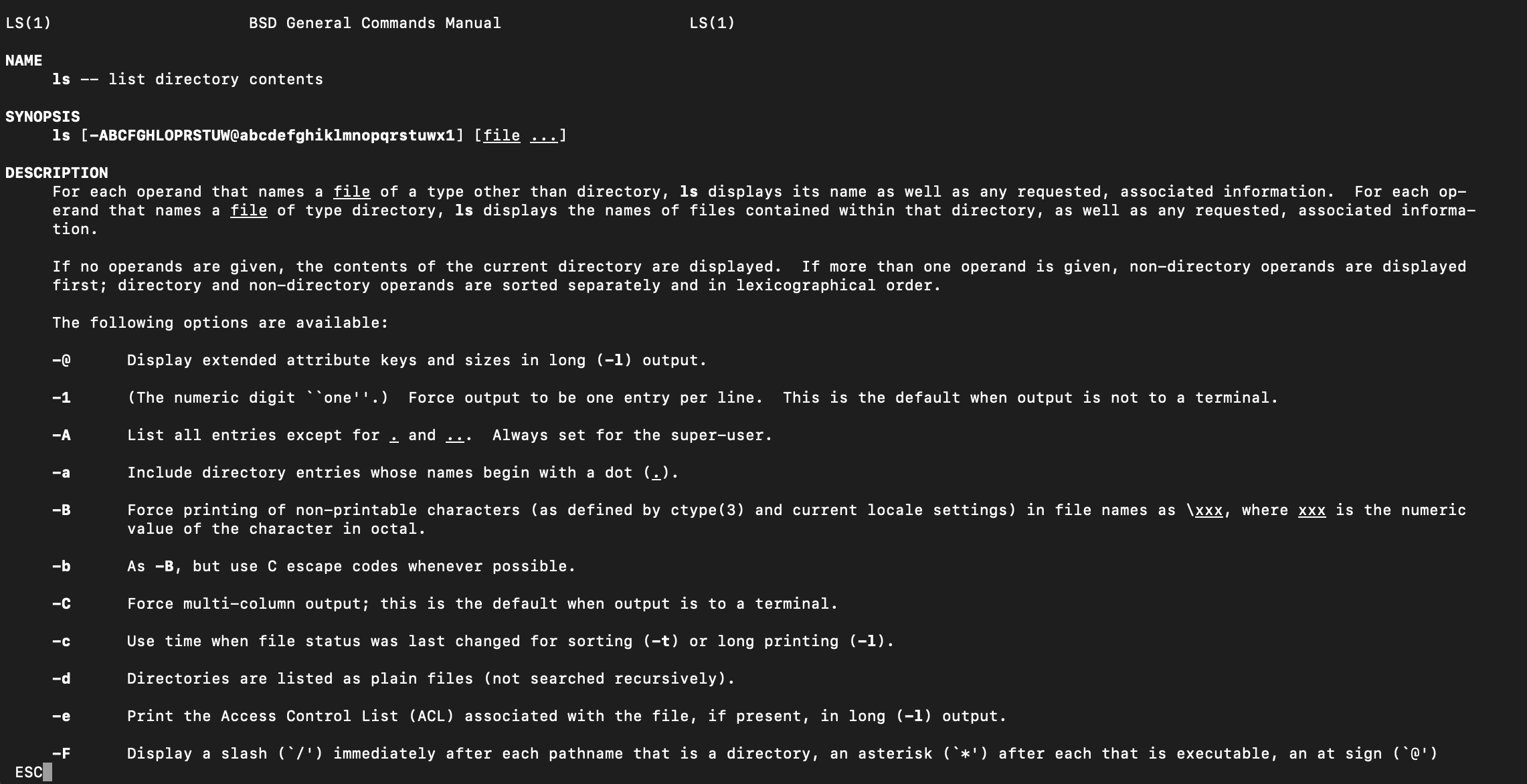


Fig. 1 `man ls`

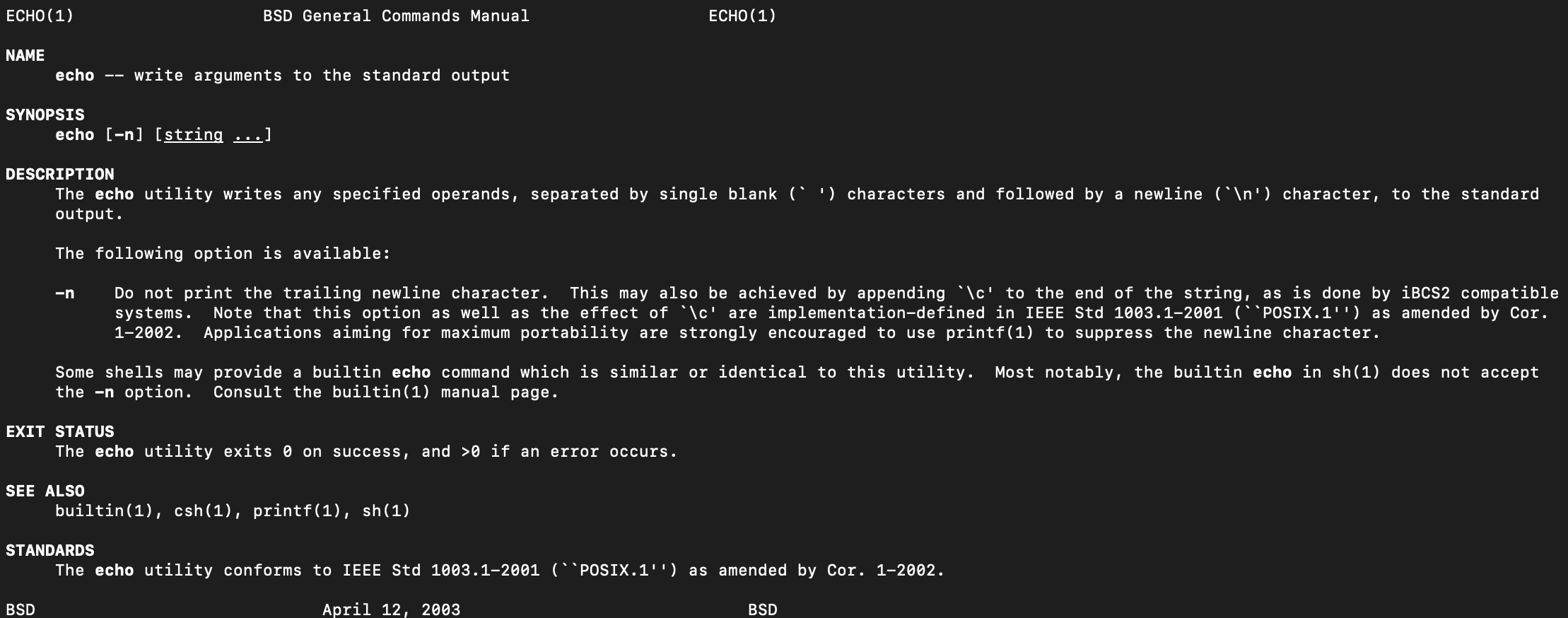


Fig. 2 `man echo`

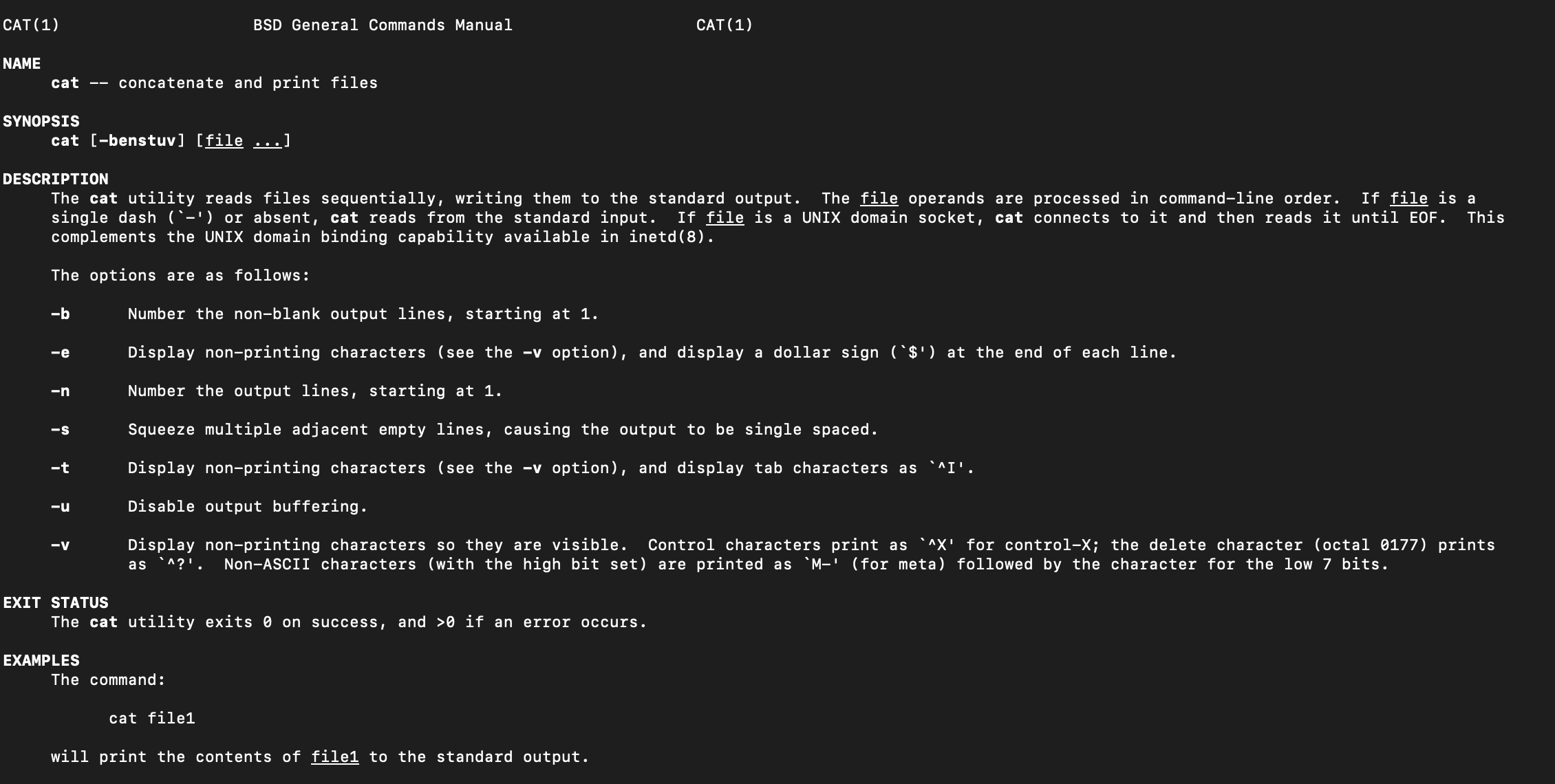


Fig. 3 `man cat`

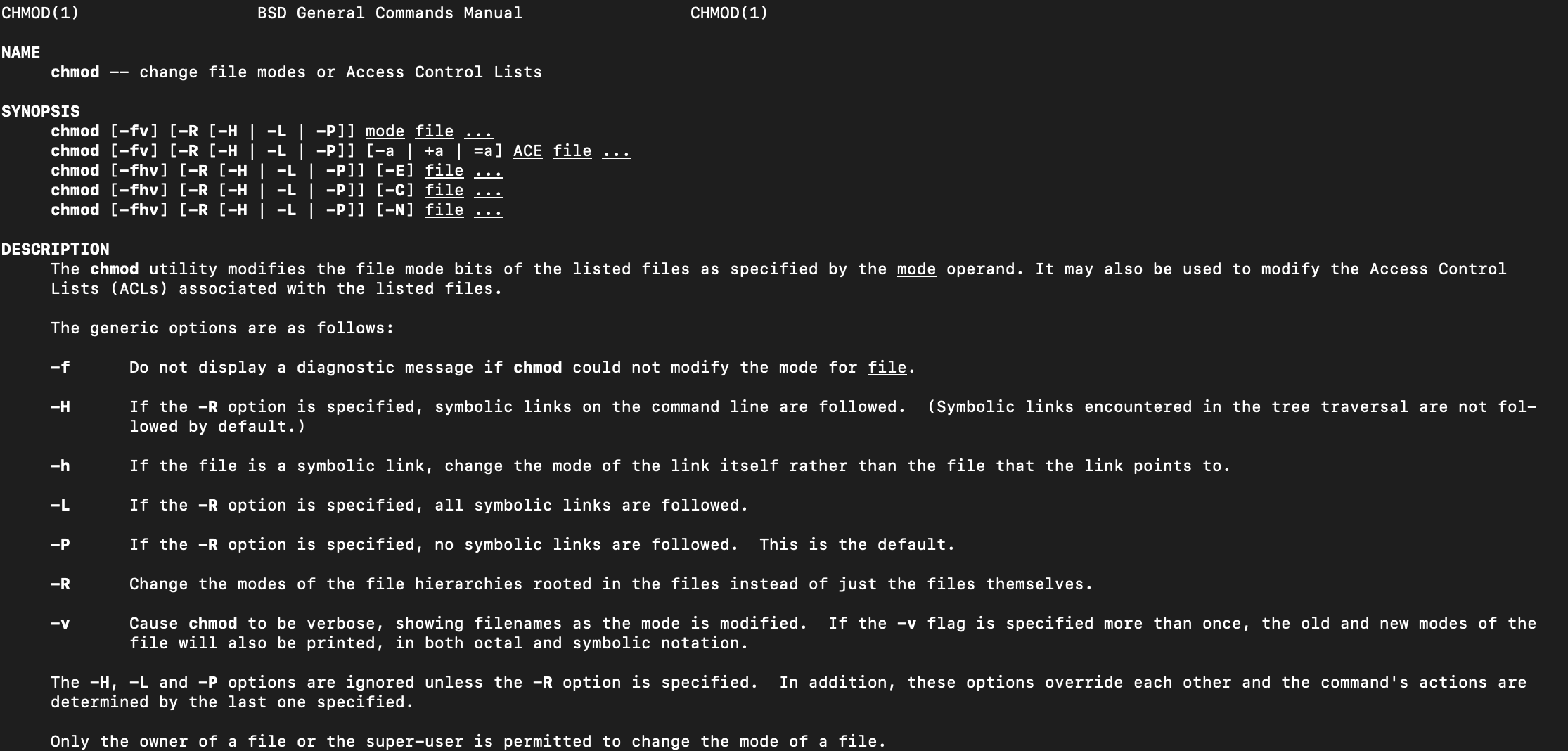


Fig. 4 `man chmod`

**Ex 2.**

**$ ls -l** (the long listing)

Lists the files in the current directory, in the long format. If the output is not piped, it also displays a total sum for all file sizes on a line before the long listing. That includes:

* File mode
* Number of links
* Owner name
* Group name
* Number of bytes in the file
* Abbreviated month, day-of-month file was last modified
* Hour file last modified
* Minute file last modified
* Pathname

**$ ls a\*b**

Lists the files that start with ‘a’ and end with ‘b’.

**$ ls -li**

-l (long listing described above)

-i inode number

**$ ls [a-z]\*[\!0-9]**

List the files in the current directory that start with **a-z** (0 or more) and don’t end with a number.

**$ ls \*[\!o]**

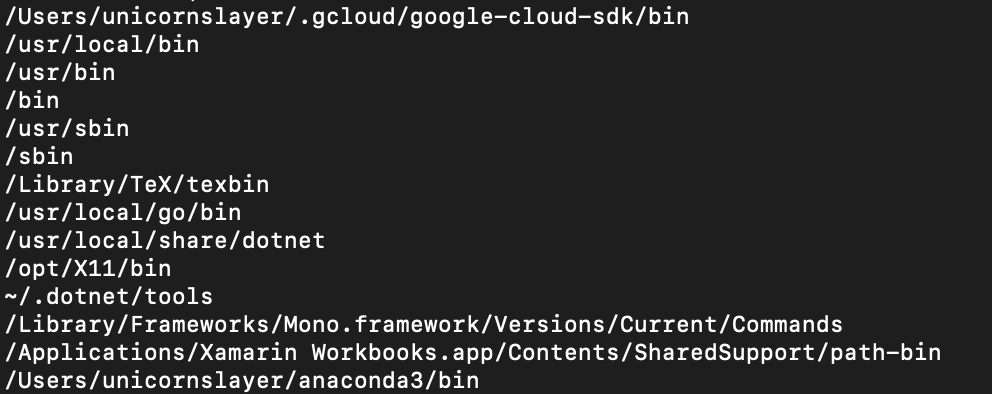
List the files that don’t end with ‘o’.

**$ ln unu doi**

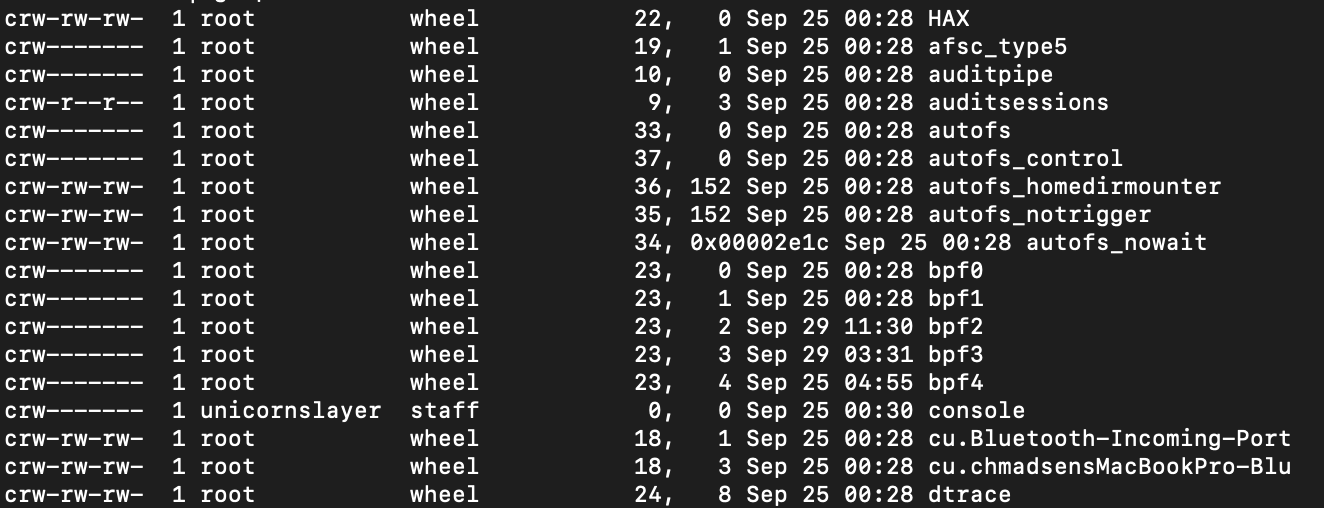
Creates a hard link from ‘unu’ to ‘doi’.

**Ex 3.**

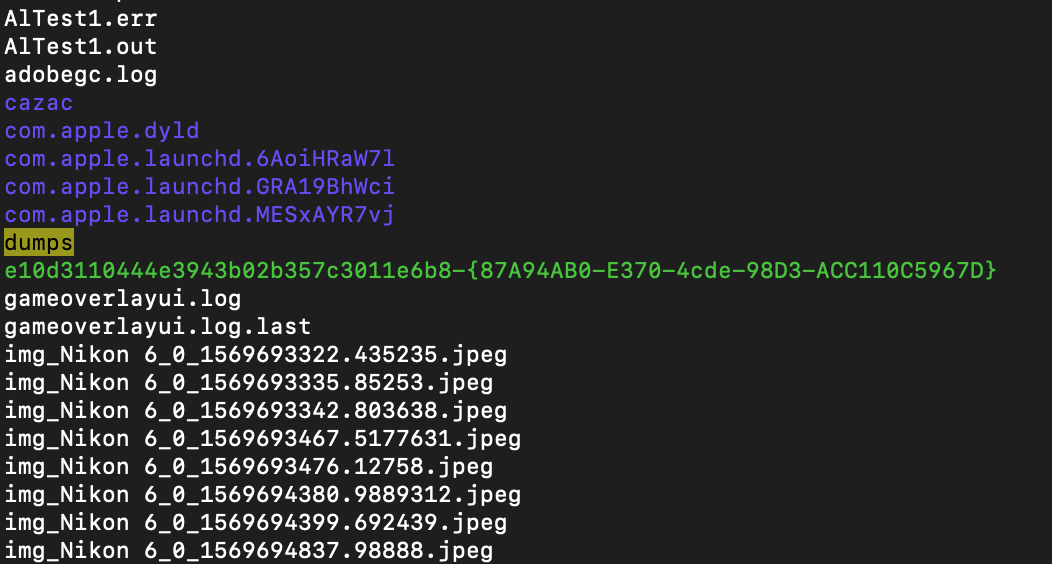
**$ echo $PATH | tr ':' '\n'** - directories with commands

****

**$ ls -l /dev | grep "^c"** - dispozitivele periferice

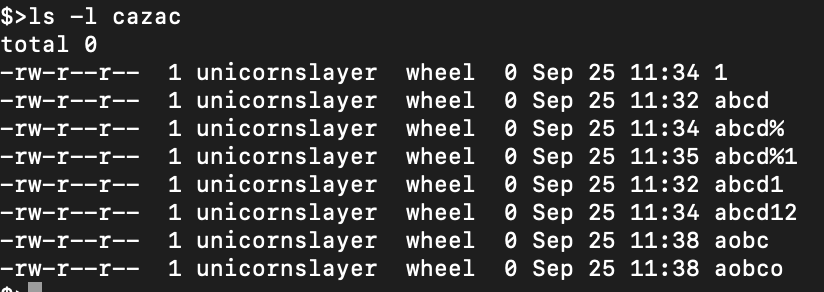


**$ ls /tmp**  - fişiere temporare



**Ex 4.**

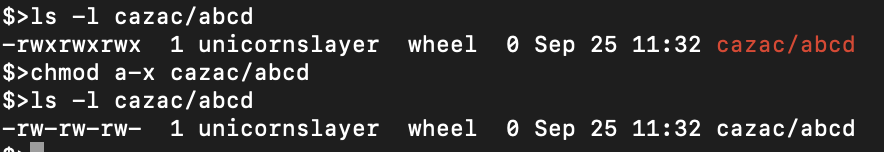
*Sa se vizulizeze continutul unui director folosind comanda ls -l.*



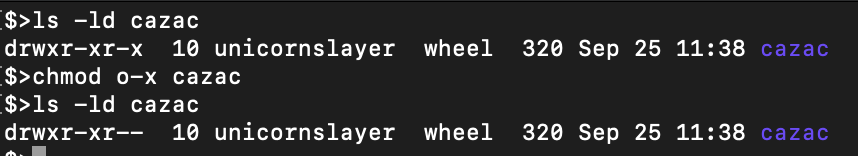
*Sa se identifice informatiile din i-node*



*Sa se modifice drepturile de acces ale unui fişier executabil* - removed exec for everyone



*Sa se modifice drepturile de acces ale unui directoriu* - removed exec for others



**Ex 5.**

*Sa se traga concluziile despre modul in care pot fi utilizate drepturile de acces pentru a permite utilizarea fişierelor in cadrul grupurilor de utilizatori.*

Group access rights are useful when we want to share a file with multiple users, but not with everyone.

**Ex 6.**

*Care este deosebirea intre un director DOS si unul UNIX ?*

* UNIX uses slashes as separators. DOS uses backslashes
* An absolute path in UNIX always starts with a slash, while in DOS, it starts with the drive
* UNIX dir max length = 255. DOS dir max length = 8 + 3 (base name + extension including the dot)

**Ex 7.**

*Ce se poate spune despre interpretarea caracterelor \* si ? in Unix. Explicati diferentele.*

In UNIX shells, ‘**\***’ and ‘**?**’ can be thought as ‘**.\***’ and ‘**.?**’ regex equivalents.

For example, to list the files starting with ‘a’ we can use: **$ ls a\***

**Ex 8.**

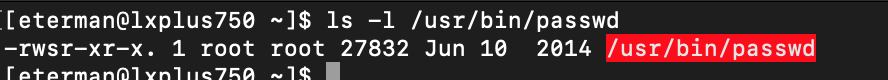
*Directorul* ***/tmp*** *este un exemplu in care bitul sticky poate fi utilizat. In acest director toata lumea are drepturi depline. In ce mod este util acest bit ?*

The sticky bit is used to prevent anyone but the owner (and the root) to delete or move the files.

**Ex 9.**

*Parola intr-un sistem cu mai multi utilizatori se schimba des pentru siguranta datelor. Fiecare dintre utilizatori poate sa-si schimbe singur parola folosind comanda passwd. Parola este pastrata intr-un fişier comun tuturor utilizatorilor, care practic este modificat de fiecare utilizator care isi schimba propria parola. Cu toate ca toti utilizatorii au acces la acest fişier ei pot sa-si modifice doar parola proprie. Sa se explice cum este realizat acest lucru.*

To change the password, the **passwd** command is usually used. Even though the **/etc/passwd** file allows only reading for normal users, everyone can change their password because the **passwd** programhas its SUID bit set. It can be seen in the first group of permissions: rw**s**



**Ex 10.**

*Din cele discutate in lucrare, s-ar parea ca daca se copiaza fişierul* ***sh*** *in directorul propriu, devenim proprietarul copiei. Daca se foloseste comanda* ***chmod*** *se pozitioneaza bitul* ***suid****, iar prin comanda* ***chown*** *se modifica proprietarul la root. Executand acum copia vom detine privilegiul de a fi root. Acest lucru nu se produce, deci care este greseala in rationament ?*

Using **strace** command to investigate what system calls it uses, we can suppose that the shell itself uses its own root powers to drop its root privileges.

But, to make it work, it’s possible to write a small program, where first **setuid(0)** is called before running the shell.

**Conclusion**

This laboratory work made me curious about SUID and why exactly it’s not working on shell.