Lista comenzilor UNIX

- **cat**
  Show the contents of a file: `cat f.txt`
- **chmod**
  change file mode bits (access read/write/execute)
  `chmod [OPTIONS] MODE FILE`

  About MODE:
  - read (r/1), write(w/2), execute (x/4)
  - Combine: read-write (rw), read-execute (rx), rwx
  - Octal: rw=1+2=3, rwx=1+2+4=7
  - Add/Remove:
    - +w (give write permission, do not change read/execute)
    - -x (remove execute permission, keep rw untouched)
    - =rw (same as rw)
  - User/Group/Other (ugo)
    - chmod 777 file (same as chmod +rwxrwxrwx)
    - chmod 731 file (same as chmod +rwxrw-r--)
    - chmod g=r (readonly for group, don't touch user/others)
    - chmod u=+x (add execution permission to user, don't touch g/o)
    - chmod g=r,o-rw (readonly for group, remove read-write from others, don't change user)

-R	recursive (files and directories in a dir)	`chmod -R +w home` # enable write for all files & dirs under home/
-f	suppress errors (--quiet)	`chmod -Rf +w home`

- **cp**
  copy files & directories
  `cp [OPTIONS] source destination`

-r, -R	recursive (files and directories in a dir)	`cp -r dir1 dir2` # copy all contents from dir1 to dir2
-s	make symbolic link instead of copying	`cp -s file_1 file_sym` # file_sym is a symlink to file_1
-l	hard link instead of copying	`cp -s file_1 file_link` # file_link is a hard link to file_1

- **cut**

  Remove sections from each line of files

  `cut OPTIONS FILE`

-c [LIST] **--characters=[LIST]**	select only these characters Use one, and only one of -b, -c or -f. Each LIST is made up of one range, or many ranges separated by commas. Selected input is written in the same order that it is read, and is written exactly once. Each range is one of: N N'th byte, character or field, counted from 1 N- from N'th byte, character or field, to end of line N-M from N'th to M'th (included) byte, character or field -M from first to M'th (included) byte, character or field Caracterele sunt date prin pozitiile lor in fisier!	`cut -c 3- list.csv` # For each line skip the first 3 characters bina,6 lena,6 ine,5 lfin,6
-f [LIST] **--fields=[LIST]**	select only these fields	`cut -f 1 list.csv` # For each line get the first field (delimited by TAB) Albina,6 Balena,6 Caine,5 Delfin,6
-d[delim] **--delimiter=[delim]**	use delim instead of TAB for field delimiter	`cut -f 2 -d , list.csv` # For each line get the second field (delimited by comma) 6 6 5 6

- **echo**

  Print message to stdout

  `echo hello`

- **expr**

  Evaluate expressions

```
   expr EXPRESSION
   expr OPTION
   Print the value of EXPRESSION to standard output.  A blank line below separates
increasing precedence groups.  EXPRESSION may be:

ARG1 | ARG2               ARG1 if it is neither null nor 0, otherwise ARG2
ARG1 & ARG2               ARG1 if neither argument is null or 0, otherwise 0
ARG1 < ARG2               ARG1 is less than ARG2
ARG1 <= ARG2              ARG1 is less than or equal to ARG2
ARG1 = ARG2               ARG1 is equal to ARG2
ARG1 != ARG2              ARG1 is unequal to ARG2
ARG1 >= ARG2              ARG1 is greater than or equal to ARG2
ARG1 > ARG2               ARG1 is greater than ARG2
ARG1 + ARG2               arithmetic sum of ARG1 and ARG2
ARG1 - ARG2               arithmetic difference of ARG1 and ARG2
ARG1 * ARG2               arithmetic product of ARG1 and ARG2
ARG1 / ARG2               arithmetic quotient of ARG1 divided by ARG2
ARG1 % ARG2               arithmetic remainder of ARG1 divided by ARG2
STRING : REGEXP           anchored pattern match of REGEXP in STRING
match STRING REGEXP       same as STRING : REGEXP
substr STRING POS LENGTH  substring of STRING, POS counted from 1
index STRING CHARS        index in STRING where any CHARS is found, or 0
length STRING             length of STRING
+ TOKEN                   interpret TOKEN as a string, even if it is a keyword like
```

```
                        'match' or an operator like '/'

( EXPRESSION )          value of EXPRESSION

Beware that many operators need to be escaped or quoted for
shells.  Comparisons are arithmetic if both ARGs are numbers,
else lexicographical.  Pattern matches return the string matched
between \( and \) or null; if \( and \) are not used, they return
the number of characters matched or 0.

Exit status is 0 if EXPRESSION is neither null nor 0, 1 if
EXPRESSION is null or 0, 2 if EXPRESSION is syntactically
invalid, and 3 if an error occurred.
```

Example:

`expr 2 + 2`	4
`expr 3 \* 2`	6
`expr 1 \> 2`	0 *fals* **A nu se scrie "expr 1 > 2", care evalueaza expr 1 si redirecteaza outputul catre file descriptor 2 (aka stderr) !!!**

`read x`
`read y`
`expr $x + $y`
	(stdin) >> 2
	(stdin) >> 3
	5

- **file**

  determine file type

```
$ file list.csv
list.csv: ASCII text
```

  (nu prea inteleg ce ar putea fi util de aici https://man7.org/linux/man-pages/man1/file.1.html)

- **find**

  search for files in a directory hierarchy

-name *pattern*	Base of file name (the path with the leading directories removed) matches shell pattern *pattern*.	`$ find -name "*.txt"`
		`./a.txt`
		`./c.txt`
		`./d.txt`

-type c	File is of type c:
	b block (buffered) special
	c character (unbuffered) special
	d directory
	p named pipe (FIFO)
	f regular file
	l symbolic link; this is never true if the -L option or the -follow option is in effect, unless the symbolic link is broken. If you want to search for symbolic links when -L is in effect, use -xtype.
	s socket
	D door (Solaris)
	To search for more than one type at once, you can supply the combined list of type letters separated by a comma `,' (GNU extension).
		`$ find -type f`
		`./2`
		`./a.txt`
		`./c.txt`
		`./d.txt`
		`./list.csv`
		`Stefan@DESKTOP-9HMTO67 ~/test`
		`$ find -type d`
		`.`
		`./some_dir`

- **grep**
  print lines that match patterns

-E	extended regex	`grep -E "a.*b" f.txt` # lines that contain ab, acb, aab etc
-i	ignore case	`grep -E -i "a.*b" f.txt` # lines that contain ab, acB, Ab etc
-v	invert match (select non matching)	`grep -E -v "^a.*$" f.txt` # lines that don't start with "a"
-c	count matching lines	`grep -E -c "a.*" f.txt` # n.o lines that start with "a"
-o	only (nonempty) matches	`grep -Eo "[0-9]+" f.txt` # all sequences of numbers in file
-q	quiet, exit status 0 if a match is found, *even if an error was detected*	`grep -Eq "abc" f.txt && echo yes` # print yes if f.txt contains "abc"
-s	suppress error messages (e.g. file not found)	`grep -Es "regex" invalid_file.txt`

-n	show line number before each match	`grep -En "ere$" d.txt`
		`1:ana are mere`
		`2:mama are pere`

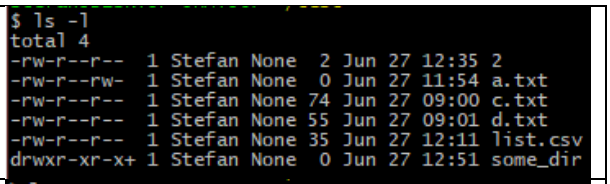
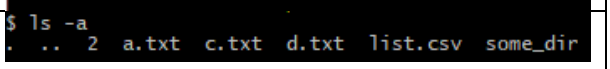
- **head**
  output the first part of files

```
-c, --bytes=[-]NUM
       print the first NUM bytes of each file; with the leading
       '-', print all but the last NUM bytes of each file

-n, --lines=[-]NUM
       print the first NUM lines instead of the first 10; with
       the leading '-', print all but the last NUM lines of each
       file

-q, --quiet, --silent
       never print headers giving file names
```

```
Stefan@DESKTOP-9HMTO67 ~/test
$ head -c 3 list.csv
Alb
Stefan@DESKTOP-9HMTO67 ~/test
$ head -n 3 list.csv
Albina,6
Balena,6
Caine,5
```

- **ls**
  list directory contents

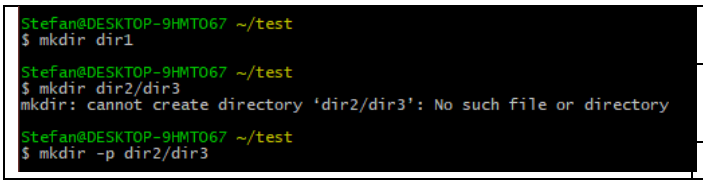
```
$ ls
2  a.txt  c.txt  d.txt  list.csv  some_dir
```

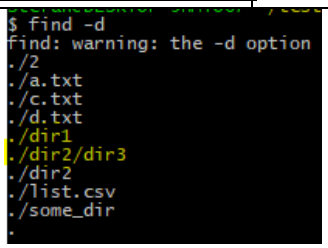
-l *(L mic)*	use a long listing format	
-a	do not ignore entries starting with .	 se poate si combo **ls -al**

- **mkdir**
  make directories

  mkdir [*OPTION*]... *DIRECTORY*...

  -p, --parents
  no error if existing, make parent directories as needed,
  with their file modes unaffected by any -m option.

	Create dir1 dir2 not found, failed with –p, dir2/dir3 is created recursively



- **mv**
  move files

  mv [*OPTION*]... [*-T*] *SOURCE DEST*
  mv [*OPTION*]... *SOURCE... DIRECTORY*
  mv [*OPTION*]... *-t DIRECTORY SOURCE...*

  -f, --force
  do not prompt before overwriting

The file rename operation can be seen as a move file command!
Rename a.txt to b.txt <=> move a.txt to b.txt

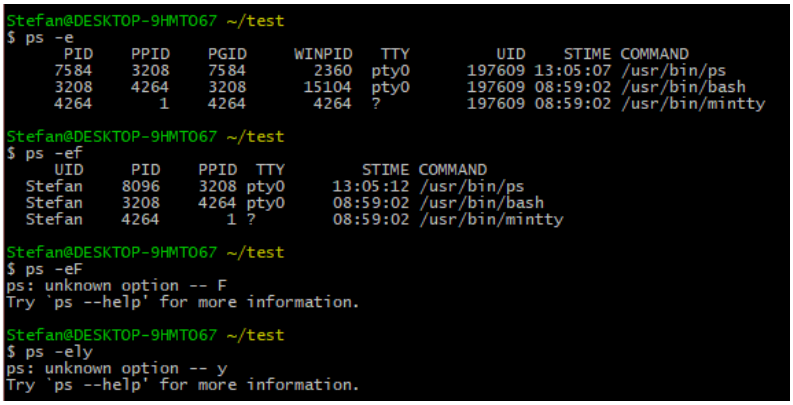
- **ps**

Report a snapshot of the current processes.

```
To see every process on the system using standard syntax:
   ps -e
   ps -ef
   ps -eF
   ps -ely
```

-F, -y nu-s in tematica.

```
To see every process running as root (real & effective ID) in
user format:
   ps -U root -u root u
```

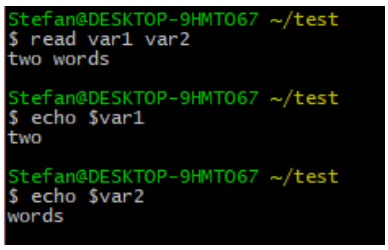



```
-f     Do full-format listing.  This option can be combined with
       many other Unix-style options to add additional columns.
       It also causes the command arguments to be printed.  When
       used with -L, the NLWP (number of threads) and LWP (thread
       ID) columns will be added.  See the c option, the format
       keyword args, and the format keyword comm.


-e     Select all processes.  Identical to -A.
```

- **read**

Read a line from the standard input ~~and split it into fields.~~

```
-p prompt output the string PROMPT without a trailing newline before
         attempting to read
```

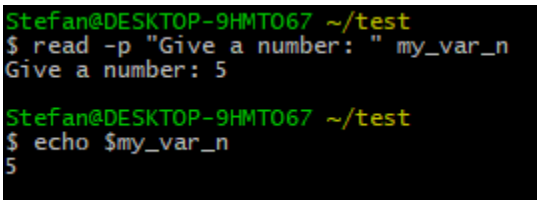


read var_name => se citeste de la stdin si continutul se poate accesa in alte comenzi cu **$var_name**

read => se citeste de la stdin si continutul se poate gasi in **$REPLY**

read var1 var2 => se pot citi mai multe variabile

read –p "message" => read cu prompt



- **rm**
  remove files or directories

  Remove (unlink) the **FILE**(s).

  -**f**, --**force**
  ignore nonexistent files, never prompt
  -**r**, -**R**, --**recursive**
  remove directories and their contents recursively

```
Stefan@DESKTOP-9HMTO67 ~/test
$ ls
2  c.txt  d.txt  dir1  dir2  list.csv  some_dir

Stefan@DESKTOP-9HMTO67 ~/test
$ rm c.txt

Stefan@DESKTOP-9HMTO67 ~/test
$ ls
2  d.txt  dir1  dir2  list.csv  some_dir

Stefan@DESKTOP-9HMTO67 ~/test
$ rm dir1
rm: cannot remove 'dir1': Is a directory

Stefan@DESKTOP-9HMTO67 ~/test
$ rm -r dir1

Stefan@DESKTOP-9HMTO67 ~/test
$ rm -rf dir2

Stefan@DESKTOP-9HMTO67 ~/test
$ ls
2  d.txt  list.csv  some_dir
```

- **sed**

stream editor for filtering and transforming text

d

Delete pattern space. Start next cycle.

s/regexp/replacement/

Attempt to match *regexp* against the pattern space. If successful, replace that portion matched with *replacement*. The *replacement* may contain the special character **&** to refer to that portion of the pattern space which matched, and the special escapes \1 through \9 to refer to the corresponding matching sub-expressions in the *regexp*.

y/source/dest/

Transliterate the characters in the pattern space which appear in *source* to the corresponding character in *dest*.

```
Stefan@DESKTOP-9HMTO67 ~/test
$ cat file.txt
ana are mere
mama are pere
vova are nuci

Stefan@DESKTOP-9HMTO67 ~/test
$ sed -E "s/are/mananca/" file.txt
ana mananca mere
mama mananca pere
vova mananca nuci
```

```
Stefan@DESKTOP-9HMTO67 ~/test
$ sed -E "s/^(\w+)/cineva (presupun ca \1)/" file.txt
cineva (presupun ca ana) are mere
cineva (presupun ca mama) are pere
cineva (presupun ca vova) are nuci
```

```
$ sed -E "y/abcdef/123456/" file.txt
1n1 1r5 m5r5
m1m1 1r5 p5r5
vov1 1r5 nu3i
```

```
Stefan@DESKTOP-9HMTO67 ~/test
$ sed -E "/mere/d" file.txt
mama are pere
vova are nuci
```

sed //d sterge **LINIILE** care dau match pe regexul specificat. NU SE STERG NUMAI MATCH-URILE!

- **sleep**
  delay for a specified amount of time (**seconds**)

  sleep 1 (1 second)
  sleep 10s (10 seconds)
  sleep 2m (2 minutes)
  sleep 1d (1 day)

- **sort**

  sort lines of text files

```
-n, --numeric-sort
       compare according to string numerical value; see manual
       for which strings are supported

-r, --reverse
       reverse the result of comparisons
```

```
Stefan@DESKTOP-9HMTO67 ~/test
$ cat words.txt
dulce
cot
descriere

Stefan@DESKTOP-9HMTO67 ~/test
$ sort words.txt
cot
descriere
dulce

Stefan@DESKTOP-9HMTO67 ~/test
$ sort -r words.txt
dulce
descriere
cot

Stefan@DESKTOP-9HMTO67 ~/test
$ while read line; do echo ${#line} $line; done < words.txt | sort -n
3 cot
5 dulce
9 descriere
```

- **tail**
  output the last part of files (opposite of **head**)

```
-c, --bytes=[+]NUM
       output the last NUM bytes; or use -c +NUM to output
       starting with byte NUM of each file
```

```
-n, --lines=[+]NUM
       output the last NUM lines, instead of the last 10; or use
       -n +NUM to skip NUM-1 lines at the start
```

```
Stefan@DESKTOP-9HMTO67 ~/test
$ tail -n 2 words.txt
cot
descriere

Stefan@DESKTOP-9HMTO67 ~/test
$ tail -c 2 words.txt
e
```

tail –c 2 numara “e” + blank character (\n sau EOF??)!!

- **test**
  check file types and compare values

```
test EXPRESSION
test
[ EXPRESSION ]
[ ]
[ OPTION


An omitted EXPRESSION defaults to false.  Otherwise, EXPRESSION
is true or false and sets exit status.  It is one of:

( EXPRESSION )
       EXPRESSION is true

! EXPRESSION
       EXPRESSION is false

EXPRESSION1 -a EXPRESSION2
       both EXPRESSION1 and EXPRESSION2 are true

EXPRESSION1 -o EXPRESSION2
       either EXPRESSION1 or EXPRESSION2 is true

-n STRING
       the length of STRING is nonzero

STRING equivalent to -n STRING

-z STRING
       the length of STRING is zero

STRING1 = STRING2
       the strings are equal

STRING1 != STRING2
       the strings are not equal
```

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

FILE1 **-ef** FILE2
       FILE1 and FILE2 have the same device and inode numbers

FILE1 **-nt** FILE2
       FILE1 is newer (modification date) than FILE2

FILE1 **-ot** FILE2
       FILE1 is older than FILE2

-b FILE
       FILE exists and is block special

-c FILE
       FILE exists and is character special

-d FILE
       FILE exists and is a directory

-e FILE
       FILE exists

-f FILE
       FILE exists and is a regular file

-g FILE
       FILE exists and is set-group-ID

-G FILE
       FILE exists and is owned by the effective group ID

-h FILE
       FILE exists and is a symbolic link (same as **-L**)

-k FILE
       FILE exists and has its sticky bit set

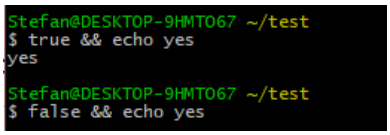
-L FILE
       FILE exists and is a symbolic link (same as **-h**)

```
-N FILE
       FILE exists and has been modified since it was last read

-O FILE
       FILE exists and is owned by the effective user ID

-p FILE
       FILE exists and is a named pipe

-r FILE
       FILE exists and the user has read access

-s FILE
       FILE exists and has a size greater than zero

-S FILE
       FILE exists and is a socket

-t FD  file descriptor FD is opened on a terminal

-u FILE
       FILE exists and its set-user-ID bit is set

-w FILE
       FILE exists and the user has write access

-x FILE
       FILE exists and the user has execute (or search) access

Binary -a and -o are ambiguous.  Use 'test EXPR1 && test EXPR2'
or 'test EXPR1 || test EXPR2' instead.
```

- **true**

do nothing, successfully

Exit with a status code indicating success (0)



lazy evaluation

- **uniq**

report or omit repeated lines

```
-c, --count
       prefix lines by the number of occurrences
```

```
Stefan@DESKTOP-9HMTO67 ~/test
$ cat a.txt
a
b
c
c
b
b

Stefan@DESKTOP-9HMTO67 ~/test
$ uniq a.txt
a
b
c
b

Stefan@DESKTOP-9HMTO67 ~/test
$ uniq -c a.txt
      1 a
      1 b
      2 c
      2 b

Stefan@DESKTOP-9HMTO67 ~/test
$ sort a.txt | uniq -c
      1 a
      3 b
      2 c
```

Liniile identice trebuie sa fie adiacente!!!

- **wc**

  print newline, word, and byte counts for each file

  -c, --bytes
  print the byte counts

  -l, --lines
  print the newline counts

  -w, --words
  print the word counts

```
Stefan@DESKTOP-9HMTO67 ~/test
$ cat file.txt
ana are mere
mama are pere
vova are nuci

Stefan@DESKTOP-9HMTO67 ~/test
$ wc -c file.txt
41 file.txt

Stefan@DESKTOP-9HMTO67 ~/test
$ wc -w file.txt
9 file.txt

Stefan@DESKTOP-9HMTO67 ~/test
$ wc -l file.txt
3 file.txt
```

- **who**

  show who is logged on

```
kbuzdar@Linux-debian:~$ who
kbuzdar  tty7         2020-06-13 09:58 (:0)
kbuzdar  pts/1        2020-06-13 09:58 (192.168.72.1)
tin      pts/2        2020-06-13 09:59 (192.168.72.1)
```