

Sisteme de operare

Tema 2

Exercițiul 1

Executați tot laboratorul pas cu pas pe mașina voastră. Folosiți paginile de manual și explicați tema propusă.

❖ Mai jos sunt toate comenzile din fișa de laborator pe care le-am executat:

```
[user@desktop-5p6viv2 ~]# echo ceva > file.txt
[user@desktop-5p6viv2 ~]# cat file.txt
ceva
```

Am pus ca redirectarea ieșirii fișierului file.txt să fie ceva.

```
[user@desktop-5p6viv2 ~]# cat file.txt | gzip > file.txt.gz
[user@desktop-5p6viv2 ~]# zcat file.txt.gz
ceva
```

Am comprimat fișierul file.txt cu comanda *gzip > file.txt.gz* și pe urmă l-am afișat cu comanda *zcat file.txt.gz*.

```
[user@desktop-5p6viv2 ~]# cat < file.txt
ceva
[user@desktop-5p6viv2 ~]# cat file.txt
ceva
```

Ambele fac același lucru, adică comanda *cat* afișează fișierul fie că are *<* sau nu.

```
[user@desktop-5p6viv2 ~]# echo "linia2" >> file.txt
[user@desktop-5p6viv2 ~]# cat file.txt
ceva
linia2
[user@desktop-5p6viv2 ~]# echo -n "linia 3" >> file.txt
[user@desktop-5p6viv2 ~]# echo "continuare" >> file.txt
[user@desktop-5p6viv2 ~]# echo "linia 4" >> file.txt
[user@desktop-5p6viv2 ~]# cat file.txt
ceva
linia2
linia 3 continuare
linia 4
```

Cu *>>* adăugăm fără a introducem terminatorul de linie, acesta fiind introdus de comanda *echo*. Dacă dorim să nu adauge linie nouă, vom pune *-n*. Putem observa mai sus cum funcționează.

```
[user@desktop-5p6viv2 ~]# echo << EOF
> salut
> hello
> EOF
```

Am început procesul de citire cu comanda *echo << EOF*, astfel, punându-i ca să se oprească când introducem EOF.

```
[user@desktop-5p6viv2 ~]# mkdir extra
[user@desktop-5p6viv2 ~]# tar -cvf extra.tar extra/
extra/
```

Cu comanda *mkdir extra* am creat folder-ul/directorul *extra* și după l-am arhivat cu comanda *tar -cvf extra.tar extra/*.

```
[user@desktop-5p6viv2 ~]# mkdir extra2
[user@desktop-5p6viv2 ~]# tar -xf extra.tar -C extra2
[user@desktop-5p6viv2 ~]# ls -lR extra2/
extra2/:
total 0
drwxr-xr-x. 2 user user 6 Mar  5 21:32 extra

extra2/extra:
total 0
```

Am creat folder-ul *extra2* ca după să dezarhivez folderul *extra* și să îl pun în *extra2* prin comanda *tar -xf extra.tar -C extra2*. După am listat *extra2* ca să vedem ce are.

```
[user@desktop-5p6viv2 ~]# tar -c extra/ | gzip > extra.tgz
[user@desktop-5p6viv2 ~]# gunzip -c extra.tgz | tar -x -C extra2/
```

```
[user@desktop-5p6viv2 ~]# tar -zcf extra.tar.gz extra/
[user@desktop-5p6viv2 ~]# tar -zxvf extra.tar.gz -C extra2/
```

Comenzile din ambele poze sunt echivalente. Prima linie arhivează și a doua linie dezarhivează.



Temă: În timp ce rulați aceste comenzi, informați-vă folosind comanda *man* despre opțiunile folosite.

```

NAME
  gzip, gunzip, zcat - compress or expand files

SYNOPSIS
  gzip [-cdhHlLnqvX] [-S suffix] [name ...]
  gunzip [-cdhHlLnqvX] [-S suffix] [name ...]
  zcat [-FAD] [-i] name ...

DESCRIPTION
  The gzip command reduces the size of the named files usingempel's coding (LZ77).
  Whenever possible, each file is replaced by one with the extension .gz, while keeping the
  same ownership modes, access and modification times. (The default extension is g for W-
  90, 95, 96, 97, 98, 99 and 00 and later.) If no files are specified, or if a file name is
  "-", the standard input is compressed to the standard output. The gzip command will only
  attempt to compress regular files. In particular, it will ignore symbolic links.

  If the compressed file name is too long for its file system, gzip truncates it. The gzip
  command attempts to truncate only the parts of the file name longer than 100 characters. A
  part is delimited by dots. If the name consists of small parts only, the longest parts
  are truncated. For example, if file names are limited to 10 characters, gzip names can be
  compressed by gunzip and zcat. Names are not truncated on systems which do not have a
  limit on file name length.

  By default, gzip keeps the original file name and timestamp in the compressed file. These
  are used when decompressing the file with the -d option. This is useful when the com-
  pressed file name was truncated or when the timestamp was not preserved after a file
  transfer.

  Compressed files can be restored to their original form using gzip -d or gunzip -o zcat.
  If the original name used in the compressed file is not suitable for its file system, a
  new name is constructed from the original one to make it legal.

  gunzip takes a list of files on its command line and replaces each file whose name ends
  with .gz, .z, .Z, or .z (ignoring case) and which begins with the correct magic num-
  bers with the original file.

  Manual page: gzip(1), gunzip(1), zcat(1), zdiff(1), zgrep(1), zmore(1)

```

gzip – comprimă un fișier sau un set de fișiere în formatul gzip și pe pune extensia .gz

gunzip – decompresă un fișier sau un set de fișiere din forma formatul gzip

-z – este o opțiune pentru unele comenzi pentru a indica faptul că fișierul trebuie să fie comprimat în format gzip

```
[user@desktop-5p6viv2 ~]$ zip -r extra.zip extra2/*
adding: extra2/extra/ (stored 0%)
[user@desktop-5p6viv2 ~]$ unzip extra.zip
Archive:  extra.zip
[user@desktop-5p6viv2 ~]$ ls -l
total 52
dr-x-----, 2 user user    6 Mar  4 16:03 dir1
dr-x-----, 2 user user   26 Mar  4 16:21 director
drwxr-xr-x, 2 user user    6 Mar  5 21:32 extra
drwxr-xr-x, 3 user user   19 Mar  5 21:36 extra2
-rw-r--r--, 1 user user 10240 Mar  5 21:33 extra.tar
-rw-r--r--, 1 user user  110 Mar  5 21:43 extra.tar.gz
-rw-r--r--, 1 user user  110 Mar  5 21:42 extra.tgz
-rw-r--r--, 1 user user  176 Mar  5 22:00 extra.zip
-rw-----, 1 user user   16 Feb 28 22:38 file1.txt
-rw-r--r--, 1 user user 10240 Mar  5 21:31 file.tar
-rw-r--r--, 1 user user    39 Mar  5 21:20 file.txt
-rw-r--r--, 1 user user   25 Mar  5 21:09 file.txt.gz
drwxr-xr-x, 2 user user   53 Mar  1 10:23 lab2_gr1
-rw-r--r--, 1 user user  355 Mar  1 10:44 problema.txt
-rw-r--r--, 1 user user    0 Mar  4 14:23 results
```

Cu comanda `zip -r extra.zip extra2/*` am arhivat folderul extra ca după să îl dezarhivez cu comanda `unzip extra.zip` și după am listat fișierele.

```
[user@desktop-5p6viv2 ~]$ cat file.txt
ceva
linia2
linia 3 continuare
linia 4
[user@desktop-5p6viv2 ~]$ sed 's/linia/coloana/' file.txt
ceva
coloana2
coloana 3 continuare
coloana 4
```

```
[user@desktop-5p6viv2 ~]$ sed 's/linia /coloana /' file.txt
ceva
linia2
coloana 3 continuare
coloana 4
```

Am folosit comanda `sed 's/linia/coloana/' file.txt` pentru a înlocui cuvântul linia cu coloanal, dar a înlocuit până și locurile unde era împreună cu alte litere așa că am pus spațiu în comandă după fiecare cuvânt.

```

[user@desktop-5p6viv2 ~]$ cat file.txt
ceva
linia2
linia 3 continuare
linia 4
linia 2LINIA 1 LINIA 3ceva
linia linia
[user@desktop-5p6viv2 ~]$ sed 's/linia /coloana /' file.txt
ceva
linia2
coloana 3 continuare
coloana 4
coloana 2LINIA 1 LINIA 3ceva
coloana linia
[user@desktop-5p6viv2 ~]$ sed 's/linia /coloana /g' file.txt
ceva
linia2
coloana 3 continuare
coloana 4
coloana 2LINIA 1 LINIA 3ceva
coloana coloana

```

Am folosit alt text în fișier pentru a arătat cazurile particulare, cum este cel în care comanda *sed 's/linia /coloana /' file.txt* care nu ia al doilea cuvânt potrivit de pe linia 6 așa că am introdus g petru a îl lua și pe acesta în considerare.

```

[user@desktop-5p6viv2 ~]$ sed 's/linia /coloana /gi' file.txt
ceva
linia2
coloana 3 continuare
coloana 4
coloana 2coloana 1 coloana 3ceva
coloana coloana

```

Am adăugat comenziile anterioare și i pentru a lua în considerare și cuvintele potrivite chair dacă acestea au litere mari, pe scurt să nu țină cont de mărimea literei.

```

[user@desktop-5p6viv2 ~]$ sed 's/linia/(&)/g' file.txt
ceva
(linia)2
(linia) 3 continuare
(linia) 4
(linia) 2LINIA 1 LINIA 3ceva
(linia) (linia)

```

Am folosit comanda *sed 's/linia/(&)/g' file.txt* pentru a marca prezența cuvântului linia între paranteze.

```

[user@desktop-5p6viv2 ~]$ sed 's/^([a-zA-Z0-9][a-zA-Z0-9]*) \([a-zA-Z0-9][a-zA-Z0-9]*)\s2 \1/ file.txt
ceva
linia2
3 linia continuare
4 linia
2LINIA linia 1 LINIA 3ceva
linia linia

```

Am inversat primele două cuvinte de pe fiecare linie.

```
[user@desktop-5p6viv2 ~]$ sed 's/([^\ ]*) \([^\ ]*\)/\2 \1/' file.txt
ceva
linia2
3 linia continuare
4 linia
2LINIA linia 1 LINIA 3ceva
linia linia
```

Am făcut comanda anterioară completă ca să ia ca cuvinte o secvență de caractere care nu conține spații.

```
[user@desktop-5p6viv2 ~]$ sed -i 's/([^\ ]*) \([^\ ]*\)/\2 \1/' file.txt
[user@desktop-5p6viv2 ~]$ cat file.txt
ceva
linia2
3 linia continuare
4 linia
2LINIA linia 1 LINIA 3ceva
linia linia
```

La ultima comanda am mai introdus -i ca să poată să facă în fișier inversarea primelor 2 cuvinte din fiecare linie.

```
[user@desktop-5p6viv2 ~]$ man grep | grep -n grep
```

```

295:      grep decides the file is a text file, it strips the CR characters from the original
296:
297:      grep understands three different versions of regular expression syntax: "basic" (BRE)
298:      "extended" (ERE) and "perl" (PCRE).  In GNU grep there is no difference in availab
299:
300:      The behavior of grep is affected by the following environment variables:
301:      grep was not compiled with national language support (NLS).  The shell command locale -
302:
303:      be concatenated with semicolons.  grep takes care of assembling the result into
304:      the language that grep uses for messages.  The default C locale uses America
305:
306:      If set, grep behaves as POSIX requires; otherwise, grep behaves more like other GR
307:
308:      Here N is grep's numeric process ID.  If the ith character of this environmen
309:
310:      variable's value is 1, do not consider the ith operand of grep to be an option
311:
312:      Email bug reports to the bug-reporting address: bug-grep@gnu.org.  An email archiv
313:
314:      https://lists.gnu.org/mailman/listinfo/bug-grep and a bug tracke
315:
316:      https://debbugs.gnu.org/cgi/pkgreport.cgi?package=grep are available.
317:      Large repetition counts in the (n,a) construct may cause grep to use lots of memory.
318:
319:      and may cause grep to run out of memory.
320:      $ grep -n -E 'foo|bar' file.txt
321:      A complete manual https://www.gnu.org/software/grep/manual/ is available.  If the inf
322:
323:      and grep program are properly installed at your site, the command
324:      info grep
325:
326: GNU grep 3.7      2019-12-29      GNU

```

Am folosit comanda *man grep | grep -n grep* pentru a afișa liniile de text, împreună cu numărul lor, unde apare cuvântul grep în pagina de manual a lui grep.

```
[user@desktop-5p6viv2 ~]$ man grep | grep -nv grep
```

```

530: addition, certain other obscure regular expressions require exponential time and space
531:
532:
533: Back-references are very slow, and may require exponential time.
534:
535: EXAMPLE
536: The following example outputs the location and contents of any line containing "T" and
537: ending in ".c", within all files in the current directory whose names contain "g" and en
538: d in ".h". The -n option outputs line numbers, the -- argument treats expansions of "g*.h"
539: starting with "." as file names not options, and the empty file /dev/null causes fil
540: names to be output even if only one file name happens to be of the form "g*.h".
541:
542: argmatch.h:1:/* definitions and prototypes for argmatch.c
543:
544: The only line that matches is line 1 of argmatch.h. Note that the regular expression syn
545: tax used in the pattern differs from the globbing syntax that the shell uses to match fil
546: names.
547:
548: SEE ALSO
549: Regular Manual Pages
550: cat(1), cmp(1), diff(1), find(1), perl(1), sed(1), sort(1), xargs(1), read(2), pcre(3)
551:
552: pcreyntax(3), pcrepattern(3), terminfo(5), glob(7), regex(7)
553:
554: Full Documentation
555:
556: should give you access to the complete manual.
557:
558:

```

Am adăugat la comanda anterioară -v pentru a afișa liniile care nu conțin cuvântul grep.

```
[user@desktop-5p6v1v2 ~]$ find / -name "*.h" | sed 's/^(.*)\./.*\1/' | sort | uniq
```

```

find: '/usr/lib/engio': Permission denied
find: '/usr/lib/private': Permission denied
find: '/usr/lib/psm': Permission denied
find: '/usr/lib/samba': Permission denied
find: '/usr/lib/audit': Permission denied
find: '/usr/lib/chrony': Permission denied
find: '/usr/cache/ldconfig': Permission denied
find: '/usr/cache/privateld': Permission denied
find: '/usr/athena': Permission denied
find: '/usr/spool/csh': Permission denied
find: '/usr/spool/csh-upload': Permission denied
find: '/usr/spool/csh': Permission denied
find: '/usr/spool/cron': Permission denied
find: '/usr/tmp/systemd-private-ch73b5d8b436f1b0896e4f344c4932e-systemd-oomd.service-7M65vT': Permi
sion denied
find: '/usr/tmp/systemd-private-ch73b5d8b436f1b0896e4f344c4932e-systemd-resolved.service-1C1Jf1': P
ermission denied
find: '/usr/tmp/systemd-private-ch73b5d8b436f1b0896e4f344c4932e-ibus-broker.service-2gP1u5': Permi
sion denied
find: '/usr/tmp/systemd-private-ch73b5d8b436f1b0896e4f344c4932e-chronyd.service-XZ11B5': Permission
denied
find: '/usr/tmp/systemd-private-ch73b5d8b436f1b0896e4f344c4932e-systemd-logind.service-c2f72p': Per
mission denied
find: '/usr/share/systemd-private-ch73b5d8b436f1b0896e4f344c4932e-MdwmManager.service-ug8Ml': Permi
sion denied
find: '/usr/share/polkitt-1/rules.d': Permission denied
find: '/usr/share/engio.nsh': Permission denied
find: '/usr/libexec/initscripts/initng-actions/auditd': Permission denied
find: '/home/user/director': Permission denied
usr/include/python3.10
usr/lib64/python3.10/site-packages/cffi
usr/lib64/python3.10/site-packages/cffi
usr/share/doc/python3-pycparser/examples/c_files
usr/share/doc/source-highlight/examples
usr/share/santitas

```

Am folosit comanda `find / -name "*.h"` pentru a căuta toate fișierele header, după am folosit comanda `sed` pentru a căuta pe fiecare linie poziția ultimului / și afișează ce este înainte lui. Am mai pus comanda `sort` ca să sorteze rezultatele și după `uniq` ca să apară o singură dată fișierul.

```

#!/bin/bash

cmdc () {
    a=$1
    b=$2
    echo "> $a $b"

    if [ $b -eq 0 ]; then
        return $a
    fi

    c=$(( $a % $b ))
    cmdc $b $c
    return $?
}

echo "parametrii sunt: $0, primul: $1, al doilea: $2"

if [ $1 -lt $2 ]; then
    echo "ordine incorecta"
    exit 1
fi

cmdc $1 $2
echo "c.m.m.d.c. este: $1"

```

Am implementat algoritmul lui Euclid într-un scrip bash.

```
[user@desktop-5p6viv2 ~]# sh fisier.txt 273 26
parametrii sunt: 273 26, primul: 273, al doilea: 26
> 273 26
> 26 13
> 13 0
c.m.m.d.c. este: 13
```

Am executat scriptul prin prima metodă.

```
[user@desktop-5p6viv2 ~]# chmod +x fisier.txt
[user@desktop-5p6viv2 ~]# ./fisier.txt 273 26
parametrii sunt: 273 26, primul: 273, al doilea: 26
> 273 26
> 26 13
> 13 0
c.m.m.d.c. este: 13
```

I-am dat mai întâi dreptul de executare și după l-am executat prin metoda a doua.

```
#!/bin/bash
cmmdc () {
    a=$1
    b=$2

    while [ ! $b -eq 0 ]; do
        c=$(( $a % $b ))
        a=$b
        b=$c
    done

    return $a
}
cmmdc $1 $2
echo "c.m.m.d.c este: $?"
```

```
[user@desktop-5p6viv2 ~]# sh fisier1.txt 273 26
c.m.m.d.c este: 13
```

Am implementat cmmdc iterativ în script și după l-am executat.

❖ Tema propusă este:



Temă: Ce părere aveți despre performanța acestor trei arhivatoare, *tar*, *gzip* și *zip*? (*tar* nu face compresie, doar concatenează).

Toate sunt arhivatoare cu performanțe bune, dar fiecare are avantaje și dezavantajele sale.

Tar face doar concatenare și nu face compresie. Dezavantajul său este că poate ocupa mai mult spațiu pe disk. Este util în situațiile în care este importantă păstrarea structurii ierarhice a fișierelor.

Gzip folosește algoritmul DEFLATE care poate comprima un fișier cu o rată de compresie bună, astfel, ocupă mai puțin spațiu pe disk. Dar poate fi mai lent procesul de compresie decât al altor algoritmi de compresie, cum sunt LZMA sau BZIP2.

Zip folosește un algoritm asemănător cu gzip, de obicei este folosit în mediul Windows. Arhivele zip pot fi utile în sistemele Linux sau macOS, dar nu sunt la fel de eficiente ca arhivele create cu gzip.

În concluzie, contează în ce scop vom folosi arhivele și mediul în care îl vom utiliza.

Exercițiul 2

Scrieți un script bash care calculează termenul Fibonacci de rang n .

```
#!/bin/bash

FibonacciRangN () {
    n=$1

    if [ $n -eq 1 ]; then
        return 1
    fi

    if [ $n -eq 2 ]; then
        return 1
    fi

    nr1=1
    nr2=1

    while [ ! $n -eq 2 ]; do
        nr3=$((nr1 + nr2))
        nr1=$nr2
        nr2=$nr3
        n=$((n - 1))
    done

    return $nr3
}

echo "rangul numarului cautat este: $1"

FibonacciRangN $1
echo "numarul este: $?"
```

```
[user@desktop-5p6viv2 ~]$ sh fisier3.txt 6
rangul numarului cautat este: 6
numarul este: 8
```

Am implementat într-un script bash un algoritm care calculează termenul Fibonacci de rang n și apoi l-am executat cu un exemplu.

Exercițiul 3

Folosiți *find* și *grep* pentru a afla ce directoare conțin fișiere cu extensia *.conf*. Scrieți un script bash care calculează suma totală a dimensiunilor directoarelor ce conțin aceste fișiere (atenție să nu aveți același director de mai multe ori). Folosiți, de exemplu, construcția cu apostrof invers ' pentru a parcurge toate liniile dintr-un fișier:

```
[user@fedora ~]$ cat some.txt
one
two
three
[user@fedora ~]$ for i in `cat some.txt`; do echo "> $i" ; done
> one
> two
> three
```

```
#!/bin/bash

readarray -t directori << (find / -type d | grep '\.config$' | sort | uniq )

total=0

for director in "${director[@]"; do
    marime=$(du -s "$director")
    total=$((total+marime))
done

echo "Dimensiunea total a directoarelor care contin fisier .config este: $total"
```

Dimensiunea total a directoarelor care contin fisier .config este: 0

Am folosit comanda *find / -type d | grep '.config\$' | sort | uniq* pentru a căuta directorii care conțin fișiere cu extensia *.config*, să le sortez și să fie unice. După am folosit cu aceasta comanda și *readarray -t directori* în care punem toți directorii găsiți din prima comandă în variabila *directori* de tip array. După am inițializat variabila *total* cu 0 și am folosit *for director in "\${director[@]"; do...* pentru a parcurge fiecare director din array-ul *directori*. În acest *for* am pus variabila *marime* care prin comanda *\$(du -s "\$director")* să ia dimensiunea directorului current și după în adună-l în *total*. La final, am afișat dimensiunea totală a directoarelor.

Exercițiul 4

Compuneți un fișier script bash care scrie 100 de linii într-un fișier cu numele *file.txt*. Liniile vor fi de forma "linia n: continut", unde "n" este de fapt numărul liniei, începând cu 1.

```
#!/bin/bash

touch file.txt

for i in {1..100}; do
    echo "linia $i: continut" >> file.txt
done
```

Am folosit linia *touch file.txt* pentru a crea fișierul file.txt. După m-am folosit de un for ca să fac 100 de linii și să le pun în fișierul file.txt așa cum trebuie să fie cu comanda *echo "linie 3: continut" >> file.txt*.

```
linia 65: continut
linia 66: continut
linia 67: continut
linia 68: continut
linia 69: continut
linia 70: continut
linia 71: continut
linia 72: continut
linia 73: continut
linia 74: continut
linia 75: continut
linia 76: continut
linia 77: continut
linia 78: continut
linia 79: continut
linia 80: continut
linia 81: continut
linia 82: continut
linia 83: continut
linia 84: continut
linia 85: continut
linia 86: continut
linia 87: continut
linia 88: continut
linia 89: continut
linia 90: continut
linia 91: continut
linia 92: continut
linia 93: continut
linia 94: continut
linia 95: continut
linia 96: continut
linia 97: continut
linia 98: continut
linia 99: continut
linia 100: continut
```

După am executat scriptul și după am afișat fișierul file.txt.

Exercițiul 5

Pentru fișierul creat la întrebarea anterioară, folosiți *sed* ca să înlocuiți toate aparițiile cuvântului "linia" cu "line", iar "continut" cu "content". Păstrați neschimbate numerele de linie, dar ștergeți ":" și înlocuiți-l cu "-". O linie de forma "linia 3: continut" se va schimba în "line 3 - content".

```
[user@desktop-5p6viv2 ~]# sed -i 's/linia /line /' file.txt
```

```
[user@desktop-5p6viv2 ~]# sed -i 's/continut/content/g' file.txt
```

```
[user@desktop-5p6viv2 ~]# sed -i 's/: / - /g' file.txt
```

```
line 65 - content
line 66 - content
line 67 - content
line 68 - content
line 69 - content
line 70 - content
line 71 - content
line 72 - content
line 73 - content
line 74 - content
line 75 - content
line 76 - content
line 77 - content
line 78 - content
line 79 - content
line 80 - content
line 81 - content
line 82 - content
line 83 - content
line 84 - content
line 85 - content
line 86 - content
line 87 - content
line 88 - content
line 89 - content
line 90 - content
line 91 - content
line 92 - content
line 93 - content
line 94 - content
line 95 - content
line 96 - content
line 97 - content
line 98 - content
line 99 - content
line 100 - content
```

Am folosit prima comandă ca să transform cuvântul linia în line, comanda a doua pentru a transforma cuvântul conținut în content și a treia comandă pentru a transforma : în -. Toate acestea realizându-se în fișier, așa că le-am afișat cu comanda *cat file.txt*.