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, punânu-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 folde-rul 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 -zxf extra.tar.gz -C extra2/
```

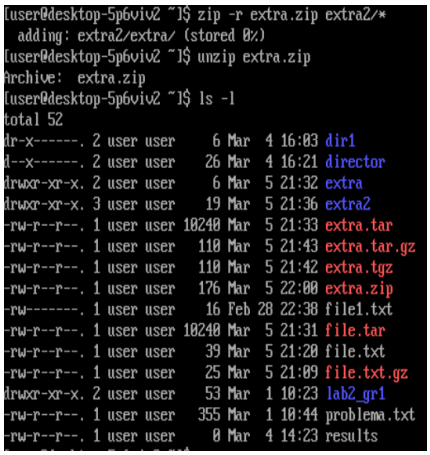
Comenzile din ambele poze sunt echivalente. Prima linie arhivează și a doua linie dezarhivează.

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

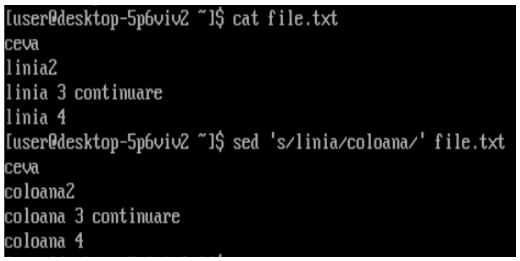
gzip – comprimă un fișier sau un set de fișiere în formatul gzip și pe pune extensia .gz

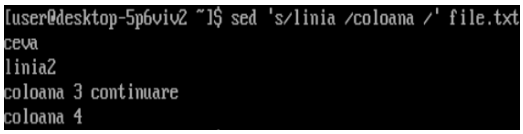
gunzip – decomprimă un fișier sau un set de fișiere din forma formatul gzip

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



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.





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 comenzii 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/\([a-zA-Z0-9][a-zA-Z0-9]*\) \([a-zA-Z0-9][a-zA-Z0-9]*\)/\2 \1/' fil
e.txt
ceva
linia2
3 linia continuare
4 linia
2LINIA linia 1 LINIA 3ceva
 linia linia
```

Am inversat primele două cuvinte de pe fiecare linie.

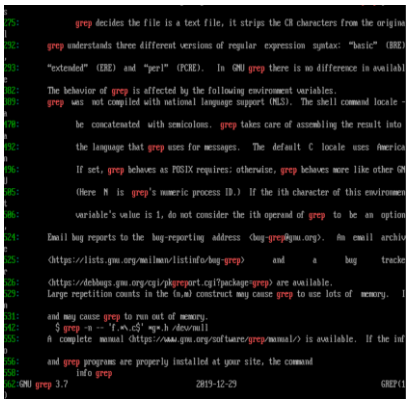
```
[user@desktop-5p6viv2 ~]$ sed 's/\([^ ][^ ]*\) \([^ ][^ ]*\)/\2 \1/' file.txt
ceva
linia2
3 linia continuare
4 linia
2LINIA linia 1 LINIA 3ceva
 linia linia
```

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.

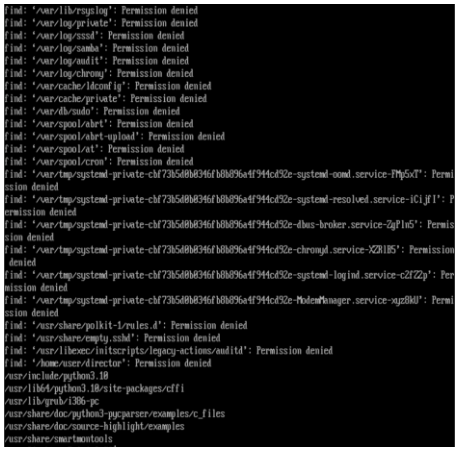
```
[user@desktop-5p6viv2 ~]$ man grep | grep -n grep
```



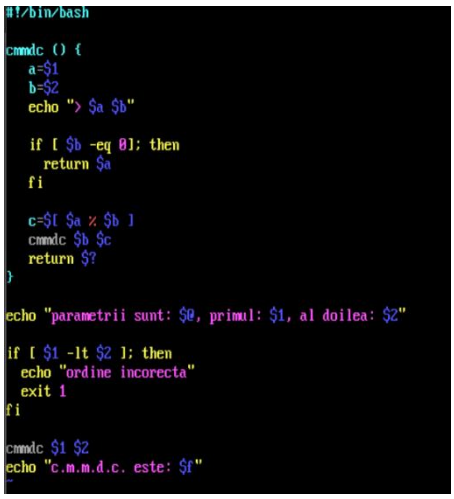
Am folosit comanda *man grep | grep -n grep* pentru a afișa liniile de text, împreună cu numărul lor, unde apare cuvântul grep în pagina de manual a lui grep.

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

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



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.



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 foloseste 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 foloseste un algoritm asemănător cu gzip, deobicei este folosit în mediul Windows. Arhivele zip pot fi utile în sistemele Linuz 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 "${directori[@]}"; do
        marime=$(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$’ | sort | uniq”* pentru a căuta directorii care conțin fișiere cu extensia .config, să le sortez și să fie unice. După am folosit cu aceasta comanda și *readarray -t directori* în care punem toți directorii gasiți din prima comandă în variabila directori de tip array. După am inițializat variabila total cu 0 și am folosit *for director in “${directori[@]}”; 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 fina, am afișat dimensiunea totala 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 sa fac 100 de linii și să le pun în fișierul file.tx așa cum trebuie să fie cu comanda *echo "linie $i: continut" >> file.txt*.

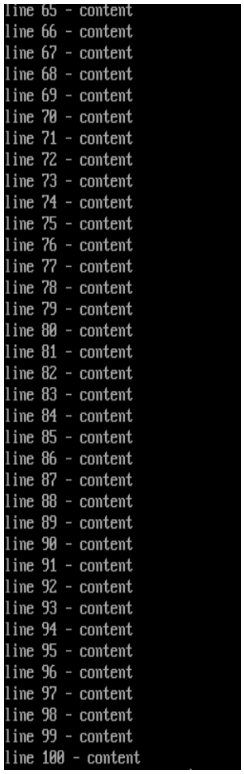
```
linia 65: continut
linia 66: continut
linia 67: continut
linia 68: continut
linia 69: continut
linia 70: continut
linia 71: continut
linia 72: continut
linia 73: continut
linia 74: continut
linia 75: continut
linia 76: continut
linia 77: continut
linia 78: continut
linia 79: continut
linia 80: continut
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
```



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