Tlemcen University
Department of Computer Science
1st Year Engineering
" Introduction to operating systems 1"

Sessions 7, 8 and 9 : Unix filters (advanced commands)

Academic year: 2023-2024

Outline

- Unix filters
  - *Modifying data in a file*
    - *Splitting a file: split*
    - *Sorting files: sort*
    - *Character string conversion: tr*
  - *Editing files with criteria*
    - *Editing a file from the end: tail*
    - *Editing a file from the beginning: head*
    - *Count the lines in a file: wc*
    - *Editing fields in a file: cut*
    - *Merging files: paste*
    - *Extraction of common lines from two files: comm*
  - *Comparing files*
    - *Compare two files: cmp*
    - *Editing differences between two files: diff*

Outline

- *grep and find commands*
  - *Regular expressions*
  - *grep command*
  - *find command*

Modifying data in a file

Splitting a file: command split

- *Split command in Linux is used to **split large files into smaller files**.*
- *The names of the files are PREFIXaa, PREFIXab, PREFIXac, and so on.*
  - *By default the PREFIX of files name is x and the default size of each split file is 1000 lines per file and both the parameters can be changed with ease*
- *It is generally used with log and archive files as they are very large.*
- *Its syntax is :*

```
split [options] name_of_file prefix_for_new_files
```

Example 1: use the split command without options

- *Assume a file name with name* ***index.txt***

```
split index.txt
```

- *We obtain the following result :*

```
mohamed@mohamed-VirtualBox:~/lesson7$ split index.txt
mohamed@mohamed-VirtualBox:~/lesson7$ ls
index.txt  xaa
mohamed@mohamed-VirtualBox:~/lesson7$ cat xaa
line 1
line 2
line 3
line 4
line 5
line 6
line 7
line 8
line 9
mohamed@mohamed-VirtualBox:~/lesson7$
```

Example 2 : use the split command with options “-l”

- *Split file based on number of lines while specifying the prefixes of the result files.*

```
split -l 4 index.txt split_file
```

- *The result is as follow :*

```
mohamed@mohamed-VirtualBox:~/lesson7$ cat index.txt
line 1
line 2
line 3
line 4
line 5
line 6
line 7
line 8
line 9
mohamed@mohamed-VirtualBox:~/lesson7$ split -l 4 index.txt split_file
mohamed@mohamed-VirtualBox:~/lesson7$ ls
index.txt  split_fileaa  split_fileab  split_fileac
mohamed@mohamed-VirtualBox:~/lesson7$ cat split_fileaa
line 1
line 2
line 3
line 4
mohamed@mohamed-VirtualBox:~/lesson7$ cat split_fileab
line 5
line 6
line 7
line 8
mohamed@mohamed-VirtualBox:~/lesson7$ cat split_fileac
line 9
mohamed@mohamed-VirtualBox:~/lesson7$
```

Example 3: Split file size using '-b' option

split -b 16 index.txt split_bytes

```
mohamed@mohamed-VirtualBox:~/lesson7$ split -b 16 index.txt split_bytes_
mohamed@mohamed-VirtualBox:~/lesson7$ ls
index.txt  split_bytes_aa  split_bytes_ab  split_bytes_ac  split_bytes_ad
mohamed@mohamed-VirtualBox:~/lesson7$ cat split_bytes_aa
line 1
line 2
limohamed@mohamed-VirtualBox:~/lesson7$ cat split_bytes_ab
ne 3
line 4
linemohamed@mohamed-VirtualBox:~/lesson7$ cat split_bytes_ac
 5
line 6
line 7mohamed@mohamed-VirtualBox:~/lesson7$ cat split_bytes_ad

line 8
line 9
mohamed@mohamed-VirtualBox:~/lesson7$
```

Example 4: Change in suffix length.

- *By default, the suffix length is 2. We can also change it using '-a' option.*

```
split -l 4 -a 4 index.txt split_
```

```
mohamed@mohamed-VirtualBox:~/lesson7$ split -l 4 -a 4 index.txt  split_
mohamed@mohamed-VirtualBox:~/lesson7$ ls
index.txt  split_aaaa  split_aaab  split_aaac
mohamed@mohamed-VirtualBox:~/lesson7$
```

Example 5: Split files created with numeric suffix.

- *In general, the output has a format of x** where ** are alphabets. We can change the split files suffix to numeric by using the '-d' option.*

```
split -l 4 -d index.txt
```

```
mohamed@mohamed-VirtualBox:~/lesson7$ split -l 4 -d index.txt
mohamed@mohamed-VirtualBox:~/lesson7$ ls
index.txt  x00  x01  x02
mohamed@mohamed-VirtualBox:~/lesson7$
```

Example 6:Create n chunks output files.

- *If we want to split a file into three chunk output files then use the '-n' option with the split command which limits the number of split output files.*

```
split -n 5 index.txt
```

```
mohamed@mohamed-VirtualBox:~/lesson7$ split -n 5 index.txt
mohamed@mohamed-VirtualBox:~/lesson7$ ls
index.txt  xaa  xab  xac  xad  xae
mohamed@mohamed-VirtualBox:~/lesson7$
```

Sorting files: command sort

- ***Command sort***
  - *SORT command is used to sort a file, arranging the records in a particular order.*
  - *By default, the sort command sorts file assuming the contents are ASCII.*
  - *Using options in the sort command can also be used to sort numerically.*
  - ***command sort** sorts the contents of a text file, line by line.*
  - *It supports sorting alphabetically, in reverse order, by number, and can also remove duplicates.*
  - *The sort command can also sort by items not at the beginning of the line, ignore case sensitivity, and return whether a file is sorted or not.*

- **The command *sort* follows these features**
  - *Lines starting with a number will appear before lines starting with a letter.*
  - *Lines starting with a letter that appears earlier in the alphabet will appear before lines starting with a letter that appears later in the alphabet.*
  - *Lines starting with a uppercase letter will appear before lines starting with the same letter in lowercase.*
- Syntax :

```
sort [options]    filename
```

❑ Example 1: sort without options

```
mohamed@mohamed-VirtualBox:~/lesson7$ cat file1.txt
kamel
fatima
ali
djamel
farid
aymen
mohamed@mohamed-VirtualBox:~/lesson7$ sort file1.txt
ali
aymen
djamel
farid
fatima
kamel
mohamed@mohamed-VirtualBox:~/lesson7$
```

Example 2: -o Option:

- *This option is used if you want to redirect the result to a new file.*

```
sort -o  file_sort.txt   file.txt
```

```
mohamed@mohamed-VirtualBox:~/lesson7$ sort -o sort_file.txt file1.txt
mohamed@mohamed-VirtualBox:~/lesson7$ ls
file1.txt  index.txt  sort_file.txt
mohamed@mohamed-VirtualBox:~/lesson7$ cat sort_file.txt
ali
aymen
djamel
farid
fatima
kamel
mohamed@mohamed-VirtualBox:~/lesson7$
```

Example 3: -r Option: Sorting In Reverse Order

- *You can perform a reverse-order sort using the "–r"* ***option***

- **Example 4: -n Option**: To sort a file **numerically**
  - *This option is used to sort the file with numeric data present inside.*

- **Example 5: sorting a table on the basis of any column number by using -k option.**
  - ***Syntax :***

```
sort -k filename.txt
```

```
mohamed@mohamed-VirtualBox:~/lesson7$ cat file2.txt
kamel 16
mohamed 12
ali 08
djamel 10
farid 14
aymen 11
mohamed@mohamed-VirtualBox:~/lesson7$ sort -k 2 file2.txt
ali 08
djamel 10
aymen 11
mohamed 12
farid 14
kamel 16
mohamed@mohamed-VirtualBox:~/lesson7$ sort -k 1 file2.txt
ali 08
aymen 11
djamel 10
farid 14
kamel 16
mohamed 12
mohamed@mohamed-VirtualBox:~/lesson7$
```

❑ Example 6: -u option: To sort and remove duplicates

```
mohamed@mohamed-VirtualBox:~/lesson7$ cat file3.txt
kamel
ali
kamel
djamel
fatima
aymen
djamel
mohamed@mohamed-VirtualBox:~/lesson7$ sort -u file3.txt
ali
aymen
djamel
fatima
kamel
mohamed@mohamed-VirtualBox:~/lesson7$
```

Character string conversion: tr ((TRanslate)

- The ***tr*** command
  - *The **tr** command is a UNIX command-line utility for converting or deleting characters.*
  - *It supports a range of transformations, including converting uppercase characters to lowercase, overwriting repetitive characters, and deleting specific characters.*
  - *It can be used with UNIX pipes to perform more complex translations.*
  - *Syntax :*

```
tr [OPTION] SET1 [SET2]
```

tr replaces each character in the input stream belonging to the SET1 with the character of the same rank in the SET2.

- In all uses of tr, the notation a z is authorized to represent the sequence of characters from a to z (inclusive) and non-printable characters are represented by their ASCII code in octal.
- Options
  - *-c: complements the set of characters in the string, i.e. the operations apply to characters not in the given set.*
  - *-d: deletes the characters in the first set from the output.*
  - *-s: replaces repeated characters listed in set1 with a single occurrence*

Example 1: convert lower case characters to upper case

- *To convert characters from lower case to upper case, you can either specify a range of characters or use the predefined character classes.*

```
mohamed@mohamed-VirtualBox:~/lesson7$ cat file4.txt
WELCOME TO
tlemcen, algeria
mohamed@mohamed-VirtualBox:~/lesson7$ cat file4.txt | tr [a-z] [A-Z]
WELCOME TO
TLEMCEN, ALGERIA
mohamed@mohamed-VirtualBox:~/lesson7$ cat file4.txt
WELCOME TO
tlemcen, algeria
mohamed@mohamed-VirtualBox:~/lesson7$ cat file4.txt | tr [:lower:] [:upper:]
WELCOME TO
TLEMCEN, ALGERIA
mohamed@mohamed-VirtualBox:~/lesson7$
```

Example 2: translate white-space characters to tabs

- *The following command translates all the white-space characters to tabs:*

```
echo "Welcome To Tlemcen" | tr [:space:] "\t"
```

Example 3: translate braces into parenthesis

```
mohamed@mohamed-VirtualBox:~/lesson7$ cat file5.txt
{Welcome to}
Tlemcen, Algeria
mohamed@mohamed-VirtualBox:~/lesson7$ tr "{}" "()" < file5.txt > outpfile.txt
mohamed@mohamed-VirtualBox:~/lesson7$ cat outpfile.txt
(Welcome to)
Tlemcen, Algeria
mohamed@mohamed-VirtualBox:~/lesson7$
```

- **Example 4: squeeze a sequence of repetitive characters using -s option.**
  - *This removes repeated instances of characters of the last SET specified*

```
mohamed@mohamed-VirtualBox:~/lesson7$ echo "Welcome      to       Tlemcen" | tr -s " "
Welcome to Tlemcen
mohamed@mohamed-VirtualBox:~/lesson7$
```

Example 5: delete specified characters using -d option

- *To delete specific characters use the -d option. This option deletes characters in the first set specified.*

```
mohamed@mohamed-VirtualBox:~/lesson7$ echo "Welcome to Tlemcen and Toronto" | tr -d T
Welcome to lemcen and oronto
mohamed@mohamed-VirtualBox:~/lesson7$
```

❑ Example 6: To remove all the digits from the string

```
mohamed@mohamed-VirtualBox:~/lesson7$ echo "My ID is 123456" | tr -d [:digit:]
My ID is
mohamed@mohamed-VirtualBox:~/lesson7$
```

Example 7: complementing sets using the -c option

- *You can complement SET1 using the -c option. For example, to remove all characters except digits, you can use the following.*

```
mohamed@mohamed-VirtualBox:~/lesson7$ echo "My ID is 123456" | tr -cd [:digit:]
123456mohamed@mohamed-VirtualBox:~/lesson7$
```

Editing files with criteria

Editing a file from the end: command tail

- *Displays the last N lines*
- *Its syntax:*

  `tail [options] [file]`

- *Displays the last 10 lines by default*

  `tail -n N <filename>` *(Displays the last N lines)*

- *Example:*

  `tail -n 5 toto.txt` *(Displays the last 5 lines of the file toto.txt)*

- *If you want to display all the lines starting with line number N*

  `tail -n +N <filename>`

  `tail -n +5 toto.txt` *or* **`tail +5 toto.txt`** *(displays all the lines in the file from the starting from the 5th line)*

- *Display several files with tail:*

  `tail -n N <file1> <file2> <file3>`

Editing a file from the beginning : *Command head*

- *Displays the first N lines*
- *Its syntax:* **`head [option] [file]`**
  - *-n: to display a specific number of lines, you use the -n followed by the number of lines.*

    ***Example**:* **`head -n 5 toto.txt`** *(Display the first 5 lines of the file toto.txt)*
- *A quicker syntax is to directly write the number of lines to display.*

  Example : **`head -5 toto.txt`**
- *Display all except the last N lines:*

  ***Example**:* **`head -n -5 toto.txt`** *(Displays all the file except the last 5 lines)*
- ***head** can display several files at once.*

Example**:* ***head -n 2 toto1.txt toto2.txt *(Displays the first two lines of each file)*

Count the lines in a file: command *wc*

- *The command **wc** counts the number of lines, words, characters and bytes in a file and returns the result.*
- *Its syntax :*

```
wc OPTION... [FICHIERs]...
```

- *The command wc accepts one or more file names.*
- *It can also be nested with other commands using a pipe.*
- *By default, the command wc displays four columns:*
  - *the number of lines,*
  - *the number of words,*
  - *the number of bytes*
  - *and the name of the file passed as an argument.*

- ***Some options:***
  - *-l, -lines: displays the number of lines*
  - *-w, -words: displays the number of words*
  - *-m, -chars : displays the number of characters*
  - *-c, -bytes : displays the number of bytes*
  - *-L, -Max-Line-Length : displays the length of the longest line*
- *If the wc command toto.txt returns the result: 12 113 415 toto.txt*
  - *12 is the number of lines*
  - *113 is the number of words*
  - *415 is the number of characters*

Editing fields in a file: command *cut*

- *The command cut can be used to cut parts of a line by **byte position, character and field**.*
- *Basically the cut command slices a line and extracts the text.*
- *It is necessary to specify option with command otherwise it gives error.*
- *If more than one file name is provided then data from each file is **not precedes** by its file name.*
- ***Its syntax:***

```
cut OPTION... [FILE]...
```

- ***Options :***
  - ***-b (bytes):*** *To extract the specific bytes, you need to follow -b option with the list of byte numbers separated by comma. Range of bytes can also be specified using the hyphen(-).*

```
mohamed@mohamed-VirtualBox:~/TP4$ cat villes.txt
Oran
Tlemcen
Algiers
mohamed@mohamed-VirtualBox:~/TP4$ cut -b 1,3,4 villes.txt
Oan
Tem
Agi
mohamed@mohamed-VirtualBox:~/TP4$ cut -b 1,3,6 villes.txt
Oa
Tee
Agr
mohamed@mohamed-VirtualBox:~/TP4$ cut -b 2-4 villes.txt
ran
lem
lgi
mohamed@mohamed-VirtualBox:~/TP4$
```

- *It uses a special form for selecting bytes from beginning upto the end of the line*
  - *2- indicate from second byte to end byte of a line*
  - *-4 indicate from 1st byte to 4th byte of a line*

- ***-c (column):*** *To cut by character.*
- *This can be a list of numbers separated comma or a range of numbers separated by hyphen(-).*
- *Its syntax is :*

```
cut -c [(k)-(n)/(k),(n)/(n)] filename
```

*where **k** denotes the starting position of the character and **n** denotes the ending position of the character in each line, if k and n are separated by "-" otherwise they are only the position of character in each line from the file taken as an input.*

- ***-f (field)*** *: is used to cut by fields rather than columns.*
- ***cut*** *uses* ***tab*** *as a default field delimiter but can also work with other delimiter by using* ***-d*** *option.*
- *Space is not considered as delimiter in UNIX.*

```
mohamed@mohamed-VirtualBox:~/TP4$ cat savants.txt
EINSTEIN : Albert : Physician
NEWTON : Isaac : Physician
LAPLACE : Pierre-Simon : Mathematician
mohamed@mohamed-VirtualBox:~/TP4$ cut -d ":" -f 1 savants.txt
EINSTEIN
NEWTON
LAPLACE
mohamed@mohamed-VirtualBox:~/TP4$ cut -d ":" -f 1,2 savants.txt
EINSTEIN : Albert
NEWTON : Isaac
LAPLACE : Pierre-Simon
mohamed@mohamed-VirtualBox:~/TP4$
```

➢ ***–complement:*** *This option can be used in the combination with other options either with* ***-f*** *or with* ***-c****.*

```
mohamed@mohamed-VirtualBox:~/TP4$ cat savants.txt
EINSTEIN : Albert : Physician
NEWTON : Isaac : Physician
LAPLACE : Pierre-Simon : Mathematician
mohamed@mohamed-VirtualBox:~/TP4$ cut --complement -d ":" -f 1 savants.txt
 Albert : Physician
 Isaac : Physician
 Pierre-Simon : Mathematician
mohamed@mohamed-VirtualBox:~/TP4$
```

Merging files: command *paste*

- *Paste command is used to join files horizontally (parallel merging) by outputting lines consisting of lines from each file specified, separated by **tab** as delimiter, to the standard output.*
- ***Its syntax :***

```
paste [OPTION]... [FILES]...
```

```
mohamed@mohamed-VirtualBox:~/TP4$ cat wilaya.txt
Tlemcen
Oran
Sidi Belabbes
mohamed@mohamed-VirtualBox:~/TP4$ cat city.txt
Maghnia
Arzew
Tlagh
mohamed@mohamed-VirtualBox:~/TP4$ cat postcode.txt
13
31
22
mohamed@mohamed-VirtualBox:~/TP4$ paste city.txt wilaya.txt postcode.txt
Maghnia Tlemcen 13
Arzew   Oran    31
Tlagh   Sidi Belabbes   22
mohamed@mohamed-VirtualBox:~/TP4$
```

❑ Options :

- ***-d (delimiter):*** *Paste command uses the tab delimiter by default for merging the files.*
  - *The delimiter can be changed to any other character by using the* ***-d*** *option.*
  - *If more than one character is specified as delimiter then paste uses it in a circular fashion for each file line separation.*

```
mohamed@mohamed-VirtualBox:~/TP4$ paste -d ":" city.txt wilaya.txt postcode.txt
Maghnia:Tlemcen:13
Arzew:Oran:31
Tlagh:Sidi Belabbes:22
mohamed@mohamed-VirtualBox:~/TP4$ paste -d ":|" city.txt wilaya.txt postcode.txt

Maghnia:Tlemcen|13
Arzew:Oran|31
Tlagh:Sidi Belabbes|22
mohamed@mohamed-VirtualBox:~/TP4$
```

- ***-s (serial):*** *this option is used to merge the files in sequentially manner.*
  - *It reads all the lines from a single file and merges all these lines into a single line with each line separated by tab. And these single lines are separated by newline.*

```
mohamed@mohamed-VirtualBox:~/TP4$ paste -s postcode.txt wilaya.txt city.txt
13      31      22
Tlemcen Oran    Sidi Belabbes
Maghnia Arzew   Tlagh
mohamed@mohamed-VirtualBox:~/TP4$
```

Extraction of common lines from two files: *comm*

- *The command **comm** compares two sorted files line by line and write to standard output; the lines that are common and the lines that are unique.*
- ***Example of use :***
  - *Suppose you have two lists of people and you are asked to find out the names available in one and not in the other, or even those common to both. In this case you can use the command **comm.***
- ***Its syntax :***

```
comm [OPTION]... FILE1 FILE2
```

- *With no OPTION used, comm produces three-column output where :*
  - *first column contains lines unique to FILE1,*
  - *second column contains lines unique to FILE2*
  - *third and last column contains lines common to both the files.*

```
mohamed@mohamed-VirtualBox:~/TP4$ cat file1.txt
Algiers
Oran
Tindouf
Tlemcen
mohamed@mohamed-VirtualBox:~/TP4$ cat file2.txt
Oran
Sidi Belabbes
Tlemcen
ville
mohamed@mohamed-VirtualBox:~/TP4$ comm file1.txt file2.txt
Algiers
                Oran
        Sidi Belabbes
Tindouf
                Tlemcen
        ville
mohamed@mohamed-VirtualBox:~/TP4$
```

Options for comm command:

-1 : *suppress first column (lines unique to first file).*

-2 : *suppress second column (lines unique to second file).*

-3 : *suppress third column (lines common to both files).*

```
mohamed@mohamed-VirtualBox:~/TP4$ comm -1 file1.txt file2.txt
        Oran
Sidi Belabbes
        Tlemcen
ville
mohamed@mohamed-VirtualBox:~/TP4$ comm -2 file1.txt file2.txt
Algiers
        Oran
Tindouf
        Tlemcen
mohamed@mohamed-VirtualBox:~/TP4$ comm -3 file1.txt file2.txt
Algiers
        Sidi Belabbes
Tindouf
        ville
mohamed@mohamed-VirtualBox:~/TP4$
```

Commands for comparing files

Compare two files: command cmp

- ***cmp*** *command is used to compare the two files byte by byte in order to find out whether the two files are identical or not.*
- *When* ***cmp*** *is used for comparison between two files, it reports the location of the first mismatch to the screen.*
- ***cmp*** *displays no message and simply returns the prompt if the files compared are identical.*
- ***Its syntax :***

```
cmp [OPTION]... FILE1 [FILE2 [SKIP1 [SKIP2]]]
```

SKIP1 and SKIP2 specify the number of bytes to skip at the beginning of each file which is zero by default

- *The command* ***cmp*** *reports the byte and line number if a difference is found.*

```
mohamed@mohamed-VirtualBox:~/TP4$ cat file11.txt
Tlemcen
Algiers
Sidi Belabbes
Mascara
mohamed@mohamed-VirtualBox:~/TP4$ cat file12.txt
Tlemcen
Algiers
Sidi Belabes
Mascara
mohamed@mohamed-VirtualBox:~/TP4$ cmp file11.txt file12.txt
file11.txt file12.txt sont différents: octet 27, ligne 3
mohamed@mohamed-VirtualBox:~/TP4$
```

Options for cmp command:

- ***-b(print-bytes)*** *: is used to display the differing bytes in the output.*

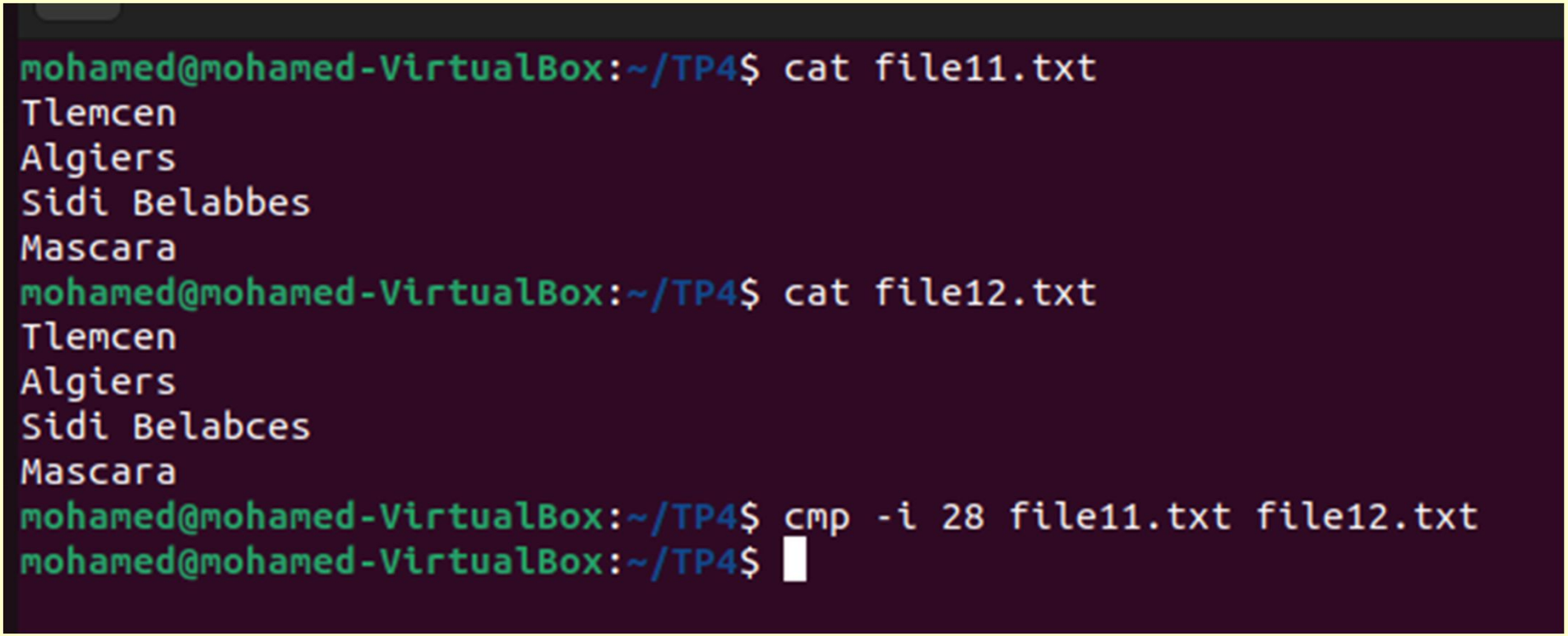
```
mohamed@mohamed-VirtualBox:~/TP4$ cat file11.txt
Tlemcen
Algiers
Sidi Belabbes
Mascara
mohamed@mohamed-VirtualBox:~/TP4$ cat file12.txt
Tlemcen
Algiers
Sidi Belabes
Mascara
mohamed@mohamed-VirtualBox:~/TP4$ cmp -b file11.txt file12.txt
file11.txt file12.txt diffèrent: octet 27, ligne 3 est 142 b 145 e
mohamed@mohamed-VirtualBox:~/TP4$
```

- ***-i [bytes-to-be-skipped] :*** *this option is used to* ***skip a particular number of initial bytes from both the files*** *and then after skipping it compares the files. This can be done by specifying the number of bytes as argument to the -i command line option.*

```
cmp –i 20 file1.txt  file2.txt
```

- ***-i [bytes to be skipped from first file] : [bytes to be skipped from second file] :****This option is very much similar to the above option but with the difference that now it* ***allows us to input the number of bytes we want to skip*** *from both the files separately.*

```
cmp –i 10:15 file1.txt  file2.txt
```


mohamed@mohamed-VirtualBox:~/TP4$ cat file11.txt
Tlemcen
Algiers
Sidi Belabbes
Mascara
mohamed@mohamed-VirtualBox:~/TP4$ cat file12.txt
Tlemcen
Algiers
Sidi Belabces
Mascara
mohamed@mohamed-VirtualBox:~/TP4$ cmp -i 28 file11.txt file12.txt
mohamed@mohamed-VirtualBox:~/TP4$

- ***-l option :*** *This option makes the cmp command print byte position and byte value for all differing bytes.*

```
mohamed@mohamed-VirtualBox:~/TP4$ cat file11.txt
Tlemcen
Algiers
Sidi Belabbes
Mascara
mohamed@mohamed-VirtualBox:~/TP4$ cat file12.txt
Tlemcen
Algiers
Sidi Belabces
Mascara
mohamed@mohamed-VirtualBox:~/TP4$ cmp -l file11.txt file12.txt
27 142 143
mohamed@mohamed-VirtualBox:~/TP4$
```

- ***-n [number of bytes to be compared] option :****This option allows you to limit the number of bytes you want to compare*

```
mohamed@mohamed-VirtualBox:~/TP4$ cat file11.txt
Tlemcen
Algiers
Sidi Belabbes
Mascara
mohamed@mohamed-VirtualBox:~/TP4$ cat file12.txt
Tlemcen
Algiers
Sidi Belabces
Mascara
mohamed@mohamed-VirtualBox:~/TP4$ cmp -n 25 file11.txt file12.txt
mohamed@mohamed-VirtualBox:~/TP4$ cmp -n 30 file11.txt file12.txt
file11.txt file12.txt sont différents: octet 27, ligne 3
mohamed@mohamed-VirtualBox:~/TP4$
```

Editing differences between two files: command diff

- *This command is used to display the differences between files by comparing them line by line, and tells us which lines in a file need to be changed to make the two files identical.*
- ***diff*** *uses certain* ***special symbols*** *and* ***instructions*** *that are required to make two files identical.*
  - ***a : add***
  - ***c : change***
  - ***d : delete***
- ***Its syntax :***

```
diff [options] File1 File2
```

```
mohamed@mohamed-VirtualBox:~/TP4$ cat a.txt
Oran
Tlemcen
Algiers
Bechar
Sidi Belabbes
mohamed@mohamed-VirtualBox:~/TP4$ cat b.txt
Tindouf
Oran
Remchi
Bechar
remchi
mohamed@mohamed-VirtualBox:~/TP4$ diff a.txt b.txt
0a1
> Tindouf
2,3c3
< Tlemcen
< Algiers
---
> Remchi
5c5
< Sidi Belabbes
---
> remchi
mohamed@mohamed-VirtualBox:~/TP4$
```

```
mohamed@mohamed-VirtualBox:~/TP4$ cat a.txt
Oran
Tlemcen
Algiers
Sidi Belabbes
Tebessa
mohamed@mohamed-VirtualBox:~/TP4$ cat b.txt
Oran
Tlemcen
Sidi Belabbes
Tebessa
mohamed@mohamed-VirtualBox:~/TP4$ diff a.txt b.txt
3d2
< Algiers
mohamed@mohamed-VirtualBox:~/TP4$
```

Some options

- ***-c (context) :*** *To view differences in context mode*

```
mohamed@mohamed-VirtualBox:~/TP4$ cat a.txt
cat
mv
comm
cp
mohamed@mohamed-VirtualBox:~/TP4$ cat b.txt
cat
cp
diff
comm
mohamed@mohamed-VirtualBox:~/TP4$ diff -c a.txt b.txt
*** a.txt       2023-11-20 11:16:50.242258715 +0100
--- b.txt       2023-11-20 11:17:57.014943065 +0100
***************
*** 1,4 ****
  cat
- mv
- comm
  cp
--- 1,4 ----
  cat
  cp
+ diff
+ comm
mohamed@mohamed-VirtualBox:~/TP4$
```

Commands for searching files and file contents

❑ *Searching files and directories : command find*

- *The command find is used to search for and locate files and directories.*
- *Searches can be carried out using different criteria (permissions, users, groups, file type, date, size, etc.).*
- *Some options :*
  - *-name: allows you to search by file name*
  - *-iname: like -name but case-sensitive*
  - *-type: allows you to filter on a type to find only files (f), only directories (d), only symbolic links (l).*
  - *-ctime, -mtime, -atime: search by creation date, last modification date or last access date.*

*-atime n: the file was last accessed n * 24 hours ago.*

find calculates the number of 24-hour periods in which the file was last accessed.

+n for more than n,

-n for less than n,

n for exactly n

- *-newer: the file has been modified more recently than another file.*
- *-size +N/-N: search by file size.*
- *-exec CMD: the file searched for which meets the above criteria and returns 0 as the output status for successful execution of the command.*
- *-ok CMD: works in the same way as -exec except that the user is prompted first.*
- *-empty: to display empty files and directories*

- *Several criteria can be combined using the following operators:*
  - *-a: AND operator*
  - *-o: OR operator*
  - *! or -not: NOT operator*

Examples

- *Search for a file by name in the current directory :*

```
find . -name toto.txt
```

- *Find all files/directories with the word toto in the name.*

```
find . -name *toto*
```

- *You can search in a specific directory such as /home/username.*

```
find /home/username -name *toto*
```

- *Search for all directories with the name home in the root directory (/).*

```
find / -type d -name home
```

- *Search several folders in parallel*

```
find ./folder1 ./folder2 -type f -name "photo*"
```

- *Find files modified in the last 10 days*

```
find / -mtime 10
```

Command grep (Global Regular Expression Print)

- ***Searching in files:*** Command **grep (Global Regular Expression Print)**
  - *The grep command searches for a string of characters in a file.*
  - *The text search pattern is called a regular expression.*
  - *When grep finds a match, it displays the line with the result.*
  - *The command grep is useful when searching large files.*

- *The main metacharacters used to form regular expressions are :*

Metacharacter	Signification
.	Generic character
^	Beginning of line
$	End of line
[...]	List of characters
[^...]	List of forbidden characters
x*	Zero, one or more occurrences of the previous element (example x)
x+	One or more occurrences of the previous element
x?	Zero or one occurrence of the previous element
\{n,m\}	At least n and at most m occurrences of the previous element
\{n,\}	At least n occurrences of the previous element
\{0,n\}	At most n occurrences of the previous element
\{n\}	Exactly n occurrences of the previous element

Examples using metacharacters

- *Generic character (.)*

```
grep o.r file.txt
Bonjour : Yes
oU : No
bonsoir : Yes
```

- *Beginning and end of string*

```
grep '^B' file.txt
grep 'r$' file.txt
grep 'ou.$' file.txt
```

- *Alternatives*

```
grep 'ou\|oi'  file.txt
```

- *Lists*

```
grep '[ji]' file.txt
grep 'n[ji]' file.txt
```

- *Intervals (Range)*

```
grep 'o[a-z]' file.txt
grep '[A-Z]o' file.txt
```

- *Repeater operators*
  - *Any occurrence of the preceding element, even if it is absent*

```
grep "bo*n" file.txt
```

  - *One or more occurrences*

```
grep "bo\+n" file.txt
```

  - *None or one occurrence*

```
grep "bo\?n" file.txt
```

```
mohamed@mohamed-VirtualBox:~/lesson8$ cat dic1.txt
bonbonbon
bonbon
booooon
boon
bno
bon
mohamed@mohamed-VirtualBox:~/lesson8$ grep "bo*n" dic1.txt
bonbonbon
bonbon
booooon
boon
bno
bon
mohamed@mohamed-VirtualBox:~/lesson8$ grep "bo\+n" dic1.txt
bonbonbon
bonbon
booooon
boon
bon
mohamed@mohamed-VirtualBox:~/lesson8$ grep "bo\?n" dic1.txt
bonbonbon
bonbon
bno
bon
mohamed@mohamed-VirtualBox:~/lesson8$
```

- *A minimum number of occurrences :*

  grep "bo\{2,\}n" file.txt

- *Exactly a number of occurrences :*

  grep "bo\{2\}n" file.txt

- *Groupings : \(\)*

  - *A grouping is used to search for a specific string and its repetition, in this case at least twice in a row:*

    ```
    grep "\(bon\)\{2\}" file.txt
    ```

```
mohamed@mohamed-VirtualBox:~/lesson8$ cat dic1.txt
bonbonbon
bonbon
booooon
boon
bno
bon
mohamed@mohamed-VirtualBox:~/lesson8$ grep "bo\{2,\}n" dic1.txt
booooon
boon
mohamed@mohamed-VirtualBox:~/lesson8$ grep "bo\{2\}n" dic1.txt
boon
mohamed@mohamed-VirtualBox:~/lesson8$ grep "\(bon\)\{2\}" dic1.txt
bonbonbon
bonbon
mohamed@mohamed-VirtualBox:~/lesson8$
```

❑ Some options for the grep command :

-A <NUM> : Display NUM lines after the searched string.

-B <NUM> : Display NUM lines before the string searched for.

-C <NUM> : Display NUM lines before and after the string searched for.

-c : Count the number of occurrences

-e : This option is used when multiple strings are searched.

-i : Do not case-sensitive

-m <NUM> : Limit the number of occurrences to NUM

-n : Display the line number in the file.

-v : Reverse match direction to select non-matching lines.

-w : Search for a whole word

-r : Recursive search

-E : Interpret the pattern as an extended regular expression.

Examples

- *To make it case-insensitive, use the -i option:*

  grep -i STUDENT * (*: in the current directory)

- *To search and print results for whole words only :*

  ***grep -w Student ***

- *To include all sub-directories in a recursive search,*

  ***grep -r Student ***

- *To display the line of a file where the occurrence is located and in all sub-directories,*

  grep -n -r Einstein /tmp/*

- *To avoid displaying lines that do not correspond to a specific character model :*

```
grep -v Teacher /tmp
```

- *grep can be used to limit the number of lines in the output :*

```
grep -m5 Student /tmp
```

- *Display a certain number of lines after the searched string :*

```
grep -A4 Student /tmp
```

- *Display a certain number of lines before the searched string :*

```
grep -B5 Student /tmp
```

- *Display a number of lines before and after the searched string :*

```
grep -C3 Student /tmp
```

- *To search for several strings at once*

```
grep -e string1 -e string2 -e string3 [file]
```

- *To search for the strings string1 or string2 in all the files in /tmp*

```
grep -r -e string1 -e string2 /tmp/*
```

or

```
grep -r -E "string1|string2" /tmp/*
```

- *To count the number of occurrences of a search string :*

```
grep -c string1 /tmp
```