TP n°5: Unix filters : advanced commands (2)

1. Objectives

The aim of this practical work is to explore some of the Unix commands that handle files: comparing files and searching for files and searching within files.

2. Unix filters

2.1 Commands for comparing files

a) **Compare two files: command *cmp***

The command ***cmp*** is used to compare the two files byte by byte to determine whether or not the two files are identical. When the command cmp is used to compare two files, it reports to the screen the position of the first different byte in the two files, and if the files being compared are identical it displays no message and simply returns the prompt.

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

Its syntax is as follows:

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

FILE1 and FILE2 are the files to be compared and SKIP1 and SKIP2 specify the number of bytes to be skipped at the beginning of each file, which is zero by default.

Options of the command ***cmp***:

- **-b(print-bytes):** is used to display the differing bytes in the output with their ASCII codes in octal.
- **-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.

  Example:

  ```
  cmp -i 10 file1 file2
  ```

  The two files file1 and file2 are compared after the 10 bytes in both files.

- **-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.

  Example:

  ```
  cmp -i 10:20 File1 File2
  ```

  In this example, we begin the comparison of the two files File1 and File2 from bytes 10 and 20 respectively.

- **-l option:** This option makes the cmp command print byte position and byte value for all differing bytes (ASCII value in octal code).
- **-n [number of bytes to be compared] option**: This option allows you to limit the number of bytes you want to compare.

  Example

  ```
  cmp -n 30 File1 File2
  ```

  This compares only the first 30 bytes in each of the two files File1 and File2.

b) Editing differences between two files: command diff

The command ***diff*** 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.

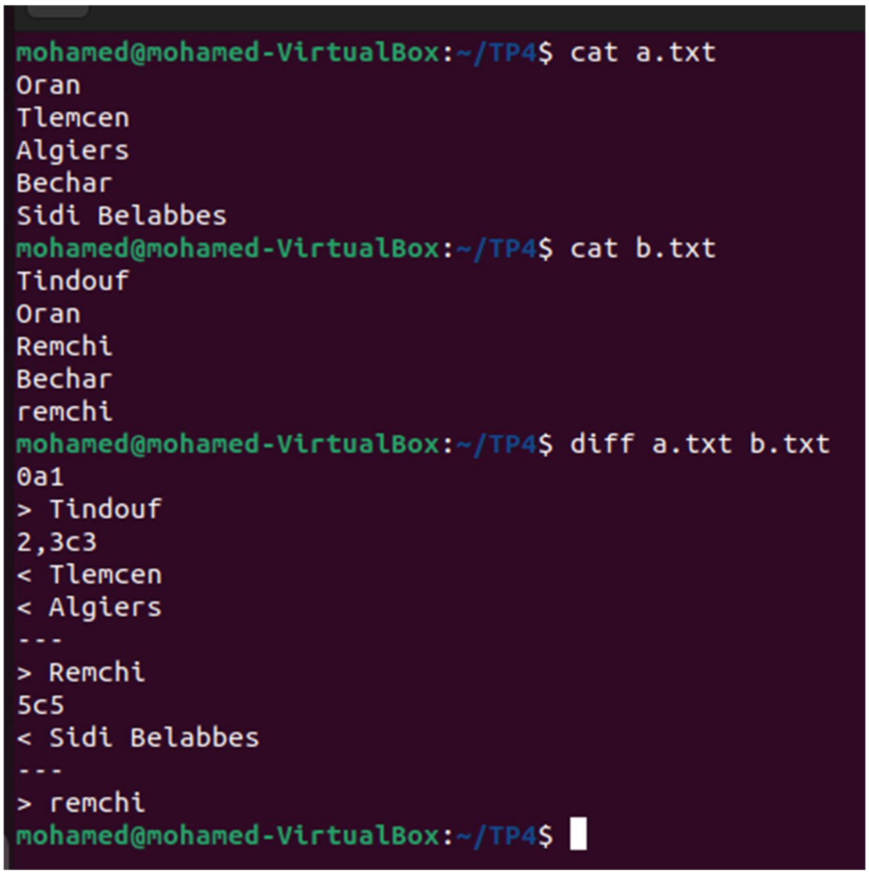
The command ***diff*** uses certain special symbols and instructions that are required to make two files identical:

- a : add
- c : change
- d : delete

Its syntax is :

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

Example :



```
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$
```

The first line of the **diff** output will contain (0a1):

- Line numbers corresponding to the first file (0),
- A special symbol (a: add) and
- Line numbers corresponding to the second file (1).

0a1 which means **after** lines 0 (at the very beginning of file) you have to add **Tindouf** to match the second file line number 1.

It indicates which lines are in each file preceded by the symbol:

- Lines preceded by < are lines from the first file.
- Lines preceded by > are lines from the second file.

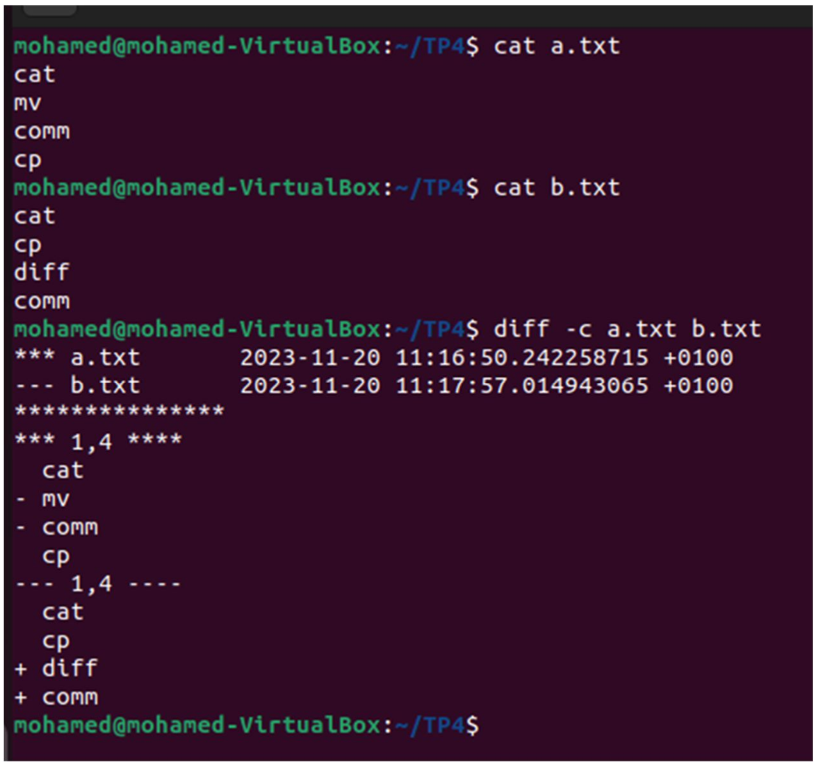
The next line contains **2,3c3**, which means that lines 2 to 3 of the first file must be modified to correspond to line 3 of the second file. It then shows us the lines containing the symbols above.

5c5: change line 5 of first file i.e. *Sidi Belabbes* with line 5 of second file i.e. Remchi.

The three dashes **("---")** merely separate the lines of file 1 (a.txt) and file 2 (b.txt).

Options of the command diff

-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$
```

The first file (a.txt) is indicated by ***, and the second file (b.txt) is indicated by ---.

The line with *************** is just a separator.

The first two lines of this output show us information about file 1 (a.txt) and file 2 (b.txt). It lists the file name, modification date, and modification time of each of our files, one per line.

The next line has three asterisks *** followed by a line range from the first file (1,4). Then four asterisks ****. After that it shows the contents of the first file with the following indicators:

(i) If the line needs to be unchanged, it is prefixed by two spaces.

(ii) If the line needs to be changed, it is prefixed by a symbol and a space. The symbol means are as follows:

+: It indicates a line in the second file that needs to be added to the first file to make them identical.

–: It indicates a line in the first file that needs to be deleted to make them identical.

2.2 Searching files and within files

a) Searching files and directories : command find

The command ***find*** can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions. By using the '-exec' other UNIX commands can be executed on files or folders found.

Its syntax is:

```
find [where to start searching from] [expression determines what to find] [-options] [what to find]
```

Options of the command ***find***:

- -exec CMD: the file searched for which meets the above criteria and returns 0 as the output state if the command is executed successfully.
- -ok CMD: works in the same way as -exec, except that the user is first prompted to do so.
- -inum N: searches for files with inode number 'N'.
- -links N : Search for files with 'N' links.
- -newer file : Search for files that were modified/created after 'file'.
- -perm octal : Search for the file if permission is 'octal'.
- -print : Display the path name of the files found by using the rest of the criteria.
- -empty : Search for empty files and directories.
- -size +N/-N : Search for files of 'N' blocks; 'N' followed by 'c'can be used to measure the size in characters; '+N' means size > 'N' blocks and '-N' means size < 'N' blocks.
- -user name : Search for files owned by username or ID 'name'.
- -type: allows you to filter on a type to obtain only files (f), directories (d) or 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

You can combine several criteria using the following operators:

- -a: AND operator
- -o: OR operator
- ! or -not: NOT operator

Examples

1. **Search for a file with a specific name.**

   ```
   find ./Folder -name file.txt
   ```

   It will search for ***file.txt*** in ***Foldr*** directory.

2. **Search for a file with pattern.**

   ```
   find ./Folder -name *.txt
   ```

   It will give all files which have '.txt' as extension.

3. **How to find and delete a file with confirmation.**

   ```
   find ./Folder -name sample.txt -exec rm -i {} \;
   ```

   When this command is entered, a prompt will come for confirmation, if you want to delete sample.txt or not. if you enter 'Y/y' it will delete the file.

4. **Search for empty files and directories.**

   ```
   find ./Folder -empty
   ```

   This command finds all empty folders and files in the entered directory or sub-directories.

5. **Search for file with entered permissions.**

   ```
   find ./Folder -perm 664
   ```

   This command finds all the files in the directory ***Folder*** or sub-directory with the given permissions.

6. **Displays repositories and sub-repositories.**

   ```
   find . -type d
   ```

   This command displays all the repositories and sub-repositories present in the current repository.

7. **Search several folders in parallel.**

   ```
   find ./Folder1 ./Folder2 -type f -name "image*"
   ```

8. **Search by modification date.**

   ```
   find / -mtime 10
   ```

   This command finds files modified in the last 10 days.

c) Searching in files: command grep

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

The 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

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
- -E : Interpret the pattern as an extended regular expression.
- -m <NUM> : Limit the number of occurrences to NUM
- -n : Display the line number in the file.
- -r : Search recursively
- -v : Reverse match direction to select non-matching lines.
- -w : Search for a whole word

Examples:

To search for a pattern in a file.

```
grep pattern filename
```

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

```
grep -i Student *
```

To search and print results for whole words only, add the **-w option**. This option only displays lines with whole-word matches and the names of the files in which they were found. For example, the word Einstein in all the files in the current directory.

```
grep -w Einstein
```

To include all sub-directories in a recursive search, add the ***-r option*** to the grep command.

```
grep -r Einstein *
```

To display the line of a file containing the occurrence, use the ***-n option***. For example, the following command searches for the word Einstein in all the sub-directories of the /tmp directory, displaying even the lines.

```
grep -n -r Einstein /tmp/*
```

We can use grep to display all lines that do not match a specific character pattern using the ***-v option***.

```
grep -v Newton /tmp
```

Log files (large files) can contain many matches for search patterns, so the grep command can be used to limit the number of lines in the output by adding the ***-m option*** and a number to the command.

```
grep -m2 Einstein /tmp
```

To display the number of lines before or after a search string, use the ***-A (After), -B (Before) and -C options.***

- -A and a certain number of lines to be displayed after the search string. For example, to display 4 lines after the word Einstein :

  ```
  grep -A4 Einstein /tmp
  ```

- -B and a certain number of lines to be displayed before the string you are looking for. For example, to display 5 lines before the word Einstein :

  ```
  grep -B5 Einstein /tmp
  ```

- -C and a certain number of lines to display before and the string you are looking for. For example, to display 3 lines before and after the word Einstein:

  ```
  grep -C3 Einstein /tmp
  ```

We can search for several words at a time using the ***-e option*** with the following syntax:

```
grep -e word1 -e word2 -e word3 [file]
```

To search for the words Einstein or Newton in all the files in /tmp, use the command :

```
grep -e Einstein -e Newton -r /tmp/*
```

or

```
grep -r -E "Einstein|Newton" /tmp/*
```

You can count the number of occurrences using the ***-c option***.

```
grep -c Einstein /tmp
```

Exercises

Exercise n° 1: command find

Use the find command to:

1) Find all files whose names begin with an uppercase or lowercase "c", possibly followed by a few letters or numbers, and end with a number.
2) List all files that meet each of the following criteria:
   a) Files that were last modified more than 10 days ago.
   b) Files that were modified 10 days ago.
   c) Files that were last modified less than 10 days ago.
3) Find files with a ".txt" extension that have been modified more recently than the "toto.txt" file.
4) Find files with the ".txt" extension that have been modified more recently than the "toto.txt" file and whose last modifications were made more than 2 days ago.
5) Find all directories from the root that have the name "etc".
6) Search for files in the user directory larger than 20 Kb ?
7) Find files in the user directory larger than 20 Kb that were modified less than 5 days ago.
8) Find all files modified between 10 and 20 days ago.
9) Display files modified in the last 5 minutes in the /home/tp directory.
10) Search for files modified within the last hour.
11) Find files whose size is > 10 MB and < 40 MB.
12) Find empty files and directories in the /tmp directory.
13) Display all directories with the name "bin" starting from the /usr directory.

Exercise n° 2 : command grep

Use the "grep" command to search in a file:

1) All lines beginning with "b".
2) All lines beginning with "B".
3) All lines beginning with "b" or "B".
4) All lines ending with "ed".
5) All lines beginning with a capital letter.
6) All lines beginning with "A", "B" or "C".
7) All lines beginning with a letter between "b" and "h".
8) All lines ending with a letter between "r" and "s".
9) All lines containing an "a" followed by an "s".
10) All lines containing an "a" not followed by an "s".
11) All lines containing an "a" not followed by an "r", "s" or "t".
12) All lines containing the string "aro" with an optional "b" between the "r" and the "o".
13) All lines not containing the string "Einstein".

14) All lines containing the string "voyage".
15) All lines containing the word "voyage".
16) All lines containing the word "voyage" and the word "voyages".
17) All lines containing the string "voyage" with the line number.
18) The number of lines containing the string "voyage".
19) All lines ending with 2 occurrences of "e".
20) The three lines preceding and following the line containing the word "voyage".
21) The five lines preceding the line containing the word "voyage".
22) The four lines following the line containing the word "voyage".