



## TP n°4: Unix filters (1)

### 1. Objectives

The aim of this practical work is to explore some of the Unix commands that handle files: modifying the contents of files, editing the contents of files, merging files and comparing their contents.

### 2. Unix filters

#### 2.1 Commands for modifying file content

##### *a) 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. This command is generally used with log and archive files as they are very large. Its syntax is as follows:

***split [options] name\_of\_file prefix\_for\_new\_files***

Options of the command split:

**-l** : this option is used to generate files with a given number of lines.

**Example:**

***split -l 10 file1.txt prefix\_file***

This command splits the file given (*file1.txt*) in parameter into files with 10 lines each except the last if the number of lines in *file1.txt* is not a multiple of 10 and the prefix of the files generated is *prefix\_file*.

**-b**: This option is used to generate files with a given number of bytes.

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

**-d**: By default the suffixes of the generated files are of the form: aa, ab, ac, ..... nevertheless with this option the suffixes of the generated files could be of this form: 01, 02, 03, .....

**-n**: this option is used to generate a certain number of output files.

**-e**: When splitting files, some output will return zero-size files. To prevent zero-size output files, use split with the -e flag.

##### *b) Sorting files: command sort*

This command is used to sort a file, arranging the records in a particular order. By default, the command *sort* sorts file assuming the contents are ASCII. It sorts the contents of a text file, line by line.

The command `sort` supports sorting alphabetically, in reverse order, by number, and can also remove duplicates. It can also sort by items not at the beginning of the line, ignore case sensitivity, and return whether a file is sorted or not. Its syntax is as follows:

**`sort [options] filename`**

Options of the command `sort`:

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

**Example:**

**`sort -o file_sort.txt file1.txt`**

where *file1.txt* is the file passed as a parameter and *file\_sort.txt* is the sorted file generated.

*-r: When you want to perform a reverse-order sort.*

*-n: This option is used to sort the file with numeric data present inside.*

*-k: This option is used to sort a table on the basis of any column number of the table.*

*-u: This option is used to sort and remove duplicates.*

### ***c) Character string conversion: command `tr`***

The command `tr` is a UNIX command-line utility for converting or removing characters. It supports a range of transformations, including converting uppercase characters to lowercase, overwriting repetitive characters, and deleting specific characters.

The command `tr` can be used with UNIX pipes to perform more complex translations. Its syntax is :

**`tr [OPTION] SET1 [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 (`\015: <return>` character).

Options of the command `tr`:

*-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*

## **2.2 Editing files with criteria**

### ***a) Editing a file from the beginning: command `head`***

The command **`head`**, as its name implies, prints the first N lines of the given input. By default, it prints the first 10 lines of the specified files. If several file names are supplied, the data for each file is preceded by its file name. It is often called with the `"-n"` option, followed by the number of lines to be displayed.

The command `head` treats empty lines as lines in their own right. Its syntax is :

***head [OPTION]... [FILE]...***

**Examples :**

- **head -n 1 <file>**: Displays the first line of the file passed as an argument.
- **head -n 100 <file>**: Displays the first 100 lines of a file.
- **head -n -N toto.txt** : Displays all lines of the file "toto.txt" except the last N lines.
- **head -n 2 toto1.txt toto2.txt** : Displays the first two lines of the files "toto1.txt" and "toto2.txt" with the names of the two files displayed.
- **head -q -n 2 toto1.txt toto2.txt** : Displays the first two lines of the files "toto1.txt" and "toto2.txt" without displaying the names of the two files.
- **head -c 20 toto.txt** : Displays the first 20 characters of the file "*toto.txt*".
- **head -c -30 toto.txt**: excludes the last 30 characters of the "*toto.txt*" file from the display.

***b) Count the lines in a file: command wc***

The command **wc** is used to find out number of lines, word count, byte and characters count in the files specified in the file arguments. By default, it displays four-columnar output wherein:

- First column shows number of lines present in a file specified,
- second column shows number of words present in the file,
- third column shows number of characters present in file,
- fourth column itself is the file name which are given as argument.

Options used in command **wc** :

- 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.

**Examples :**

If the command **wc 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

**wc toto.txt toto1.txt** : this command returns the following result:

```
14 112 403 toto1.txt
5 58 275 toto2.txt
19 170 678 total
```

**wc -l /etc/passwd**: displays the number of lines in file /etc/passwd.

### c) Editing fields in a file: command cut

The command *cut* in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by *byte position*, *character* and *field*.

The command **cut** cuts a line and extracts the text. The option must be specified with the command, otherwise an error will occur. Its syntax is :

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

Options of the command *cut*:

- **b**: 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 (-). It is necessary to specify list of byte numbers otherwise it gives error. Tabs and backspaces are treated like as a character of 1 byte.

#### Example :

The file cities.txt contains the following data:

```
Oran
Tlemcen
Algiers
```

**cut -b 1,4 cities.txt** : this command gives the following result:

```
On
Tm
Ai
```

**cut -b 2-4 cities.txt** : this command gives the following result:

```
ran
lem
lgi
```

It uses a special form for selecting bytes from beginning upto the end of the line:

#### Example :

**cut -b -3 cities.txt** (-3 indicate from 1st byte to 3rd byte of a line)

The result of this command is :

```
Ora
Tle
Alg
```

**-c:** To cut by character use the **-c** option. This can be a list of numbers separated comma or a range of numbers separated by hyphen(-). Tabs and backspaces are treated as a character. It is necessary to specify list of character numbers otherwise it gives error with this option. The syntax used :

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

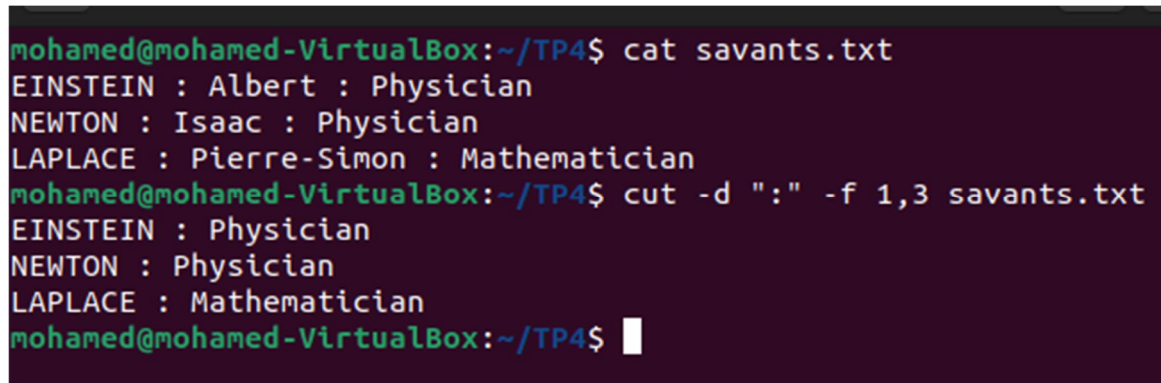
where **k** denotes the starting position of the character and **n** denotes the ending position of the character in each line

**-f (field):** To extract the useful information, you need to split by field rather than by column. The list of specified field numbers must be separated by a comma. Ranges are not described using the **-f** option. cut uses tabs as the default field delimiter, but can also use other delimiters using the **-d** option. Its syntax is as follows:

```
cut -d "delimiter" -f (field number) file.txt
```

A space is not considered a delimiter under UNIX.

**Example :**



```
mohamed@mohamed-VirtualBox:~/TP4$ cat savants.txt
EINSTEIN : Albert : Physician
NEWTON : Isaac : Physician
LAPLACE : Pierre-Simon : Mathematician
mohamed@mohamed-VirtualBox:~/TP4$ cut -d ":" -f 1,3 savants.txt
EINSTEIN : Physician
NEWTON : Physician
LAPLACE : Mathematician
mohamed@mohamed-VirtualBox:~/TP4$
```

In this example, we display fields 1 and 3.

**–complement:** As the name suggests it complement the output. This option can be used in the combination with other options either with **-f** or with **-c**.

**Example :**

```
cut --complement -c 4 cities.txt
```

```
Ora
Tlece
Algers
```

With this option, this command displays all characters except the one occupying the 4th position

#### ***d) 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. When no file

is specified, or put dash (“-”) instead of file name, paste reads from standard input and gives output as it is until a interrupt command [Ctrl-c] is given. Its syntax is :

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

### Example:

Let's consider three files named *city*, *wilaya* and *postcode*. The *city* and *wilaya* file contain 3 city and wilaya names respectively. The postcode file contains 3 numbers.

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

Options of the command *paste* :

**-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.

### Example :

```
paste -d "|" postcode.txt city.txt wilaya.txt
```

**-s (serial):** We can merge the files in sequentially manner using the **-s** option. 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 city.txt wilaya.txt
13      31      22
Maghnia Arzew   Tlagh
Tlemcen Oran    Sidi Belabbes
mohamed@mohamed-VirtualBox:~/TP4$
```

### e) Extraction of common lines from two files: command comm

The command comm compares two files sorted line by line and writes the common and unique lines to the standard output.

Suppose you have two lists of people and you need to find the names that are in one and not the other, or even those that are common to both. The command `comm` is the command that will help you do this. It requires two sorted files which it compares line by line. Its syntax is :

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

- If no `OPTION` is used, the command `comm` produces output in three columns,
  - the first of which contains the lines specific to `FILE1`,
  - the second the lines specific to `FILE2`,
  - the third and last the lines common to both files.
- The `comm` command only works if you are comparing two files which have already been sorted.

### **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).

## Exercises

### Exercise n° 1

- 1) Write the command to replace "a" with "A", "," with ";" and "/" with "\_" in *file1.txt* and redirect the result of this command to *file2.txt*.
- 2) Write the command to replace \$ with F, () with {} and <tab> with <space> in *file1.txt* and redirect the result of this command to *file2.txt* (<tab> is 011 in octal).
- 3) Write the command that converts uppercase letters to lowercase letters of the file *file1.txt* and redirect the result of this command to *file2.txt*.
- 4) Write the command to replace several consecutive spaces with a single space in *file1.txt* and redirect the result to *file2.txt*.
- 5) Write the command that replaces several consecutive <new line> with a single one in *file1.txt* and redirects the result to *file2.txt*.
- 6) Write the command that replaces several consecutive spaces or <new line> or <tab> with a single one in the file *file1.txt* and redirects the result to the file *file2.txt*.
- 7) Write the appropriate command to build an alphabetical dictionary of all the words in the file *file1.txt* and sort them alphabetically, eliminating duplicates (the generating dictionary is called *file\_dic*).

### Exercise n° 2

- 1) Write the commands that split the file *file1.txt* into files with 5 lines and the prefix "Ing", and check the number of lines in each of the files created.
- 2) Write the commands that divide the file *file1.txt* into 5 files with a suffix whose length is 3, and return the number of lines in each of the files created.
- 3) Write the command that splits a file into 5 files with a numeric suffix, avoiding the creation of files of zero size.
- 4) Write the commands that divide a file into files each containing 8 characters and check the number of characters in each of the files created. What do you notice?

### Exercise n° 3

- 1) Display the number of lines, characters and words in the "/etc/passwd" file.
- 2) Display the lines in the "/etc/passwd" file that meet each of the following criteria:
  - a) the first 10 lines,
  - b) the last 10 lines,
  - c) all lines from the tenth line onwards,
  - d) lines between 10 to 25.
- 3) Retrieve lines 4 to 10 from a file of 15 lines using pipes (this operation can be performed in two ways).
- 4) How do you display the tenth line of a file?
- 5) Create the file "test.txt" after executing the following commands:  

```
cat > test.txt
```



first tp

second tp

third tp

fourth tp

fifth tp

6) Display the contents of the file "test.txt" sorted in reverse alphabetical order.