

Computer Systems Linux II



Contents

1	Users and groups	2
1.1	Kind of users	2
1.1.1	Configuration files	3
2	Links: hard Links and symbolic Links	3
3	Wildcards	4
4	Commands	6
4.1	Commands for managing files and directories	6
4.2	Commands for managing content of files	10
4.3	Regular expressions	16
4.4	grep command	18
4.5	find command	20

1 Users and groups

In Linux, users are identified by a user number called UID (User ID). That number is different from any other UID number or other user. It is like the ID card number. That number identifies just one person and it can't be used again for another live person. Besides, each user should belong to a group, called primary groups. Moreover, the same user can belong to other groups called secondary groups:

- The primary group in most cases, has the same name as your login name. The primary group is used by default when creating new files (or directories), modifying files or executing commands.
- Secondary groups: these are groups that you are a member of outside of your primary group. It means that if some user belongs to some group as a secondary, that user will have the same (permission or accessibility) applied to the group of the file or directory.

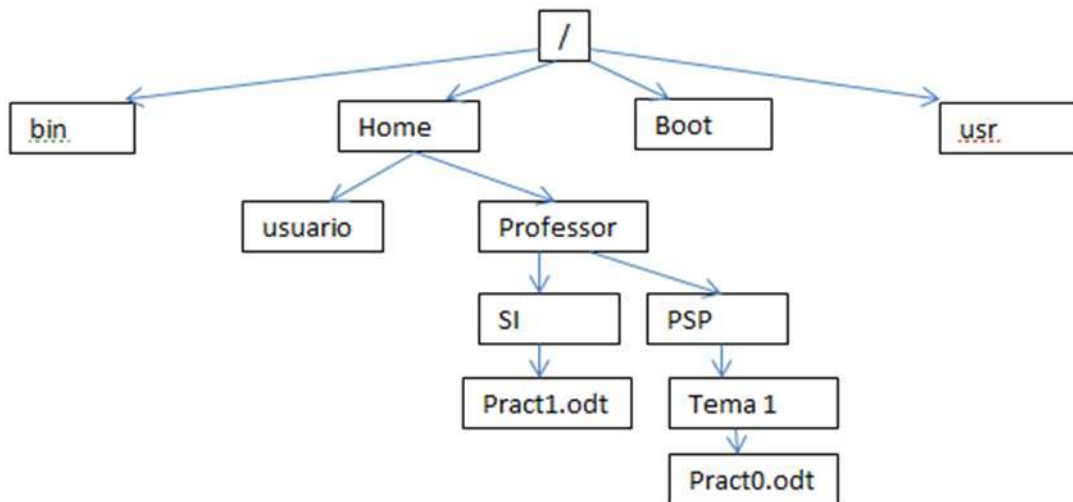
A group is a set of users, but it is possible that a group has just one user. In the same way that user has a UID number, groups are identified by GID (Group ID). As you might have guessed, it is a different number for each group.

1.1 Kind of users

There are three different kinds of users in the Linux system:

- **Root user:** also known as superuser or administrator. It is the user that has all of the privileges, and it can do almost everything. It has total access of the system, and it is in charge of the administration, updating and maintenance; Its UID number is 0.
- **System users:** They are generated when the operating systems are installed or when some service is installed. Although they do not have all the privileges, they have some of them depending on the user. They aren't used in order to validate in a session, but they are necessary in order to execute some services or processes in the operating system. The number of UID is greater than 1 and lower than 1000, except the nobody user. The UID for that user is 65534 which is the last.
- **Regular users:** They are normal users which will connect to the system and they can start a session in the operating system. They have a work directory in the /home directory. They have all the permission in that directory. The UID assigned to these kinds of users is greater or equal to 1000, although it is possible to change it. When the operating system is installed, one of these kinds of user is created by default. A regular user has just access to the files of its property or those files or directories with the permission which somebody has given to it.

1.1.1 Configuration files



It is known that in the Linux directories system, every folder is born from «/» slash or root. In order to use relative and absolute paths some special characters are used:

- “.” \Rightarrow points out the current folder
- “..” \Rightarrow points out the parent folder.

In the last picture, the path for the file pract1.odt would be:

- Absolute path: /home/Professor/SI/Pract1.odt
- Relative path for the following cases as example:
 - If the current folder is PSP \Rightarrow ../SI/Pract1.odt
 - If the current folder is “/” \Rightarrow /home/Professor/SI/Pract1.odt
 - If the current folder is usuario \Rightarrow ../Professor/Pract1.odt

Try it 1

Specify the relative and absolute path for the file Pract0.odt if the current folder is:

- bin
- SI
- “/”

2 Links: hard Links and symbolic Links

It is known that files are objects used in order to save data. But in Linux we can distinguish between two different kinds of files: regular files and links. Links are references to other files or directories in the form of an absolute or relative path, and that affects pathname resolution. It is like a shortcut. There are two kind of links in Linux:

- Hard link: the changes carried out in the link file are reflected in the original one and vice versa. Both files share the same i-node so it means that the reference is the same hard-disk area. If some of them are removed, the other will remain with the information. Besides, a file can have more than one link.

It will be possible to make a hard link if the file which is needed to make the link, is in the same partition of the hard-disk. The reason for this is because each partition has its i-node table. On the other hand, it is not allowed to create a hard link to the folders.

Hard links benefits:

- Make incremental backups saving space and time in the process.
 - When a file is copied from one place to another place, it is possible to make a hard-disk in order to save time and space.
 - To classify information.
- Symbolic link: it is a faster shortcut to some file from another place. In this case, the symbolic link saves only information about the source file. In case the file is removed, the link would lose all the information.

Symbolic link properties:

- Any change introduced in source file or in the symbolic link is assimilated for both.
- If source file is removed, the file is completely erased; and it would be impossible to access it.
- If symbolic linked file is erased, the source file will not be erased, so it will be possible to access the information through the source file.
- It is possible to create symbolic links of folders.
- The symbolic link can be used in any place, partition and file system.
- If the placement of the source file is changed, the link is broken

Symbolic link benefits:

- In order to extend file systems. It is possible to make symbolic links to another file system. For instance, if it is working in ext4 file system, it is possible to make symbolic links in order to read or write files in NTFS.
- When a huge file is copied to another placement, it is wasting a lot of time. However, when a symbolic link is possible it saves a lot more time.

It is possible to acquire more information through this link:

<https://www.youtube.com/watch?v=a000kNxDJ3c>

3 Wildcards

Wildcard characters are used in order to replace a character or a group of them. They are used in some commands related with files; for example, the commands which list or search files and directories in Linux. The wildcards that are possible to use:

- * (asterisk): that special character refers to a string of characters of any size even 0 size. For instance:

- a^* refers every word which starts with the letter a and the $*$ replaces any character or characters after the letter a; for example, arbol, as, asimetria, etc...
- $*a^*$ refers to every word in which a is contained because the first $*$ replaces any character or group of them and the $*$ after a is the same. For example: calculo, transporte, etc...
- ? (question mark) replaces just one character in the same position it is placed in the pattern. For example:
 - $h?r$: the coincidences will be those words with any letter in the second place with “h” before and r after it. After that string of a group of characters, 0 or 1 will be placed. For example: hora, horonabilidad, har, etc...
- [] (brackets): specifies just one character of the list inside of the brackets. Examples:
 - $[abc]$: the specified character can be a, b or c
 - $[.,_]$: character “.” , “,” or “_”
 - $[a - z]$: any lowercase letter
 - $[0 - 9]$: any number
 - $[a - zA - Z]$: any lowercase and uppercase letter
- [!] (Exclamation mark inside of the brackets): when the exclamation mark is placed inside of the brackets, refers to any character which is not in the list in that position when it is placed inside of the pattern. Examples:
 - $[!0 - 9]$: represents any character except numbers.
 - $[!a - zA - Z]$: represents any character except lowercase and uppercase letters.
 - $[0 - 9!]$:it refers to any number or exclamation mark because that exclamation mark is not the first character of the group.
- {} (braces): braces are used in order to point out some string of characters, at least 2 characters. Each string of characters should be listed with “,”. Examples:
 - $Pe\{lota,lotero,lazo\}$: the words which have the string lota, lotero or lazo. For example: pelota, pelotero and pelotazo. Don’t use spaces inside of the braces.



Try it 2

1. Build the wildcard pattern in order to find the following words:

- (a) acme, acme2; acme23a
- (b) smartphone92; smartphoneAB;
- (c) logo4, logo3F; logoFFF

2. What group of words will find the wildcard pattern?

- (a) $\text{logo}[ABC]$
 - i. logoA; logoB; logoABC
 - ii. logoAB;logoA;logoB
 - iii. logoA;logoB;logoC

- (b) `logo[!ABC]`
- i. `logoD; logo4; logo3`
 - ii. `logoA; logoB; logoC`
 - iii. `logoD; logo4; logoDL`

4 Commands

4.1 Commands for managing files and directories

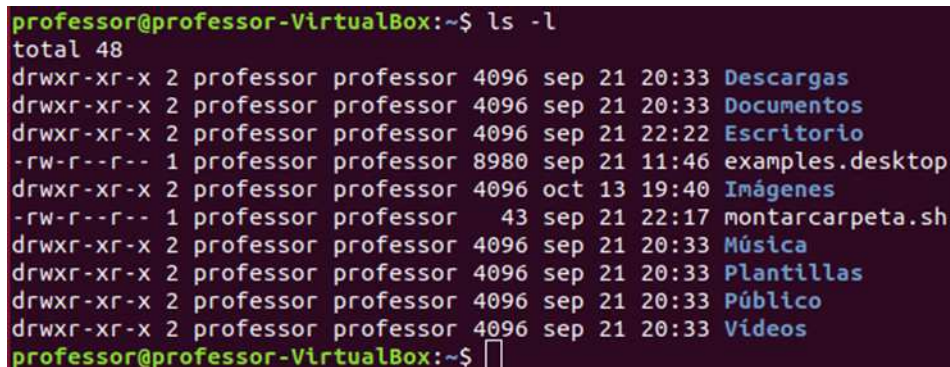
- **ls** (list) \Rightarrow show us information about files and directories. If nothing is specified, information about current directory is displayed.

Syntax:

`ls [options] [arguments]`

Options:

- `-d` \Rightarrow `--directory`; show information about the directory instead of the content of it.
- `-a` \Rightarrow `--all`; lets us see the hidden files. The name of those files starts with “.”
- `-A` \Rightarrow `--almost-all`; lists all files and directories, except the directories “.” and “..”
 - * “.” is the current directory (it refers to itself)
 - * “..” is the parent directory (it refers to the directory above)
- `-l` \Rightarrow it shows extensive information about the directories and file at the current directory



```
professor@professor-VirtualBox:~$ ls -l
total 48
drwxr-xr-x 2 professor professor 4096 sep 21 20:33 Descargas
drwxr-xr-x 2 professor professor 4096 sep 21 20:33 Documentos
drwxr-xr-x 2 professor professor 4096 sep 21 22:22 Escritorio
-rw-r--r-- 1 professor professor 8980 sep 21 11:46 ejemplos.desktop
drwxr-xr-x 2 professor professor 4096 oct 13 19:40 Imágenes
-rw-r--r-- 1 professor professor 43 sep 21 22:17 montarcarpeta.sh
drwxr-xr-x 2 professor professor 4096 sep 21 20:33 Música
drwxr-xr-x 2 professor professor 4096 sep 21 20:33 Plantillas
drwxr-xr-x 2 professor professor 4096 sep 21 20:33 Público
drwxr-xr-x 2 professor professor 4096 sep 21 20:33 Vídeos
professor@professor-VirtualBox:~$
```

The last image shows how the information got through `ls -l` command. As it can be seen, the information is shown by columns. Each row shows the information of each file or directory of the current one. The information is:

- * First column: shows the permissions of each directory or file. (the next unit is more detailed with information involving more about permissions). Directories are those rows which are started by letter “d” and files are those rows which starts with “-”
- * Next column informs about how many directories or files are contained inside of it.
- * Third column informs about the owner of that file or directory
- * Fourth column shows information about the owner group

- * Next is information about the size
 - * Next 3 columns show us information about the date and the hour of modification of that file or directory.
 - * The last column informs about the file or directory name.
- `-h` \Rightarrow `--human` –readable; this argument is used with “l” or “s” arguments. It is used in order to get information about the size of each file or directory with the greatest unit of measure.



Try it 3

Compare the information you got with `ls -l` and `ls -lh`. Pay attention to the size of each file or directory.

- `-i` \Rightarrow `--inode`; it shows the i-node file number
- `-n` \Rightarrow `--numericuid-gid`; the same result with `-l` argument; but this time, name and group owner are substituted by GID and UID.



GID and UID

GID and UID are group number and identification user number. They will be explained in the next unit.

- `-c` \Rightarrow information will appear organized by creation date.
- `-t` \Rightarrow information will appear organized by modification date.
- `-r` \Rightarrow `--reverse`; information listed is organized by reverse order.



Mix options

It is possible to mix some arguments at the same time.

Compare the following commands:

1. `ls -lc`
2. `ls -lcr`

- `-color` \Rightarrow it shows the information with color.
- `-F` \Rightarrow `--classify`; it shows information about the kind of file. The symbols which appear with the name mean:
- * * executable
 - * / directory
 - * @ symbolic link
 - * | pipe
 - * no symbol points out regular file
- `-R` \Rightarrow `--recursive`; shown the directories under the current one. Recursive way
- `-s` \Rightarrow `--size`; blocks size file
- `-S` \Rightarrow files are ordered by size.

Try it 4

1. From your personal directory (/home/user_name), show the large information of tty0 and tty9 files. These files are in /dev directories. Do it with wildcards. Just one command.
2. Show files of /dev directory which start with tty, and afterwards with any characters.
3. Show files and directories of your personal directory. Show the i-node and information about what kind of file they are.
4. Show the large information of your personal directory.
5. Show the same information with the greatest measure of unit possible.

Help

It is possible to get command information through terminal with:

```
command_name --help  
ls --help
```

- **pwd** (print working directory) \Rightarrow Show the absolute path of the current or working directory.
- **mkdir** (make directory) \Rightarrow command to create directories.

Options:

- -p: create parent directories.

For example: to create the following structure: /p1/p2/p3

`mkdir -p /p1/p2/p3` \Rightarrow p1, p2 and p3 will be created if they don't exist.

- -v: to confirm that the directory is created.

- **cd** (change directory) \Rightarrow command in order to change directory. It is possible to use some options:

- “-”: to change where the last directory was
- “..” : to change the parent directory. That is to say, the directory above of the current one
- “~”: to change to the personal directory (home).

Arguments \Rightarrow in order to change to some other directory, it is needed to specify through which one, relative or absolute path. For instance, follow the image of the first point of this unit. If the current directory is usuario, then it is required to go to Tema1 directory:

- Relative path: `cd ../Professor/PSP/Tema1`
- Absolute path: `cd /home/Professor/PSP/Tema1`

Try it 5

1. Execute the following commands:

- (a) Show the absolute path of the working directory
- (b) Go to the root directory
- (c) Go to the /etc/init.d directory
- (d) Come back to the last directory without using any path in the argument.
- (e) Check where you are
- (f) Go to your personal directory.
- (g) Check out that “.” And “..” are hard links of the current and parent directory.

- **rmdir** (remove directory) \Rightarrow erase empty directories.
- **rm** (remove) \Rightarrow remove files and directories. It is possible to use some of the following options:
 - **-f** \Rightarrow dismiss files which don't exist.
 - **-i** \Rightarrow it ask before removing each directory or file
 - **-r** \Rightarrow **-R** \Rightarrow erase the directory, erasing first every file, directory and subdirectory inside it in a recursive way
 - **-v** \Rightarrow show a message of each file or idrectory erased.
- **cp** (copy) \Rightarrow copy one or more files to another file or directory. Options:
 - **-f** \Rightarrow if the destination exists and it is not possible to open, erase the destination and try to copy it again.
 - **-i** \Rightarrow ask before overwriting.
 - **-R** \Rightarrow copy directories and the directories under it. The structure is copied.
- **mv** (move) \Rightarrow this command is to move files to another file or directory. It is the same, cut and paste. It is used to rename files.

Syntax:

```
mv [options] source destination
```

Example:

```
mv myfile.txt myfiles
```

Options:

- **-i** \Rightarrow the user is asked before overwriting
- **-f** \Rightarrow enforce writing
- **-v** \Rightarrow a message is showed for each moved file.

- **file** \Rightarrow show the type of the file.

Syntax:

```
file [filename]
```

- **du** (disk usage) \Rightarrow it shows the file size.

Syntax:

```
du -options [filename]
```

Options:

- **-b** \Rightarrow it shows the size file in bytes.
 - **-h** \Rightarrow it shows the file size in the greatest measure unit.
- **df** (display free) \Rightarrow it shows free space in storage devices. Options:
 - **-h** \Rightarrow it shows the size in the greatest measure unit
 - **-a** \Rightarrow all of the files are shown, even those files with size 0. includes duplicate and inaccessible file systems.

4.2 Commands for managing content of files

- **cat** (catenate) \Rightarrow the content of the file will be shown.

Syntax:

```
cat -options [filename]
```

Options:

- **-n** \Rightarrow number of the lines will be shown
- **head** \Rightarrow the 10 first lines of the file will be shown.

Syntax:

```
head -options [filename]
```

Options:

- **-n** \Rightarrow it shows the n first lines instead of 10 first lines.
- **tail** \Rightarrow the last 10 lines of the file will be shown.

Syntax:

```
tail -options [filename]
```

Options:

- **-n** \Rightarrow it shows the n last lines instead of 10 last lines.
- **wc** \Rightarrow it shows number of lines, words, characters and bytes which are indicated, or by the size of the largest line. Options:

- -c ⇒ number of bytes
- -m ⇒ number of characters
- -l ⇒ number of lines
- -w ⇒ number of words.
- -L ⇒ size of the largest line file.
- **more** ⇒ the paginated file is shown. That command is used for a large file which shows up. It needs more than the space of one screen. The key “q” will be used in order to quit the file, and the space bar will be pushed in order to move to the next page.
- **less** ⇒ the same from that of the last command. To surf on the file with the cursor keys is permitted.
- **sort** ⇒ it shows in ascendant order the content file by the first letter of each line.

Syntax:

```
sort [options] [filename]
```

Options:

- -c ⇒ that option checks out whether the content file is sorted. If not a message with the first line which is not sorted is shown. That option doesn’t sort the file
- -r ⇒ sort in reverse direction.
- -u ⇒ erase the repeated lines
- -n ⇒ the content is sorted numerically.
- -t ⇒ field delimiter.
- -k? ⇒ where ? is the name of the field. Normally it is used with -t option.

For example, for the following file called file0.txt:

```
professor@professor-VirtualBox:~$ cat file0.txt
rose 1234
redsell 2345
lilypink 3456
madmax 4567
nolives 5678
lilkington 6789
sutton 4444
bint 2222
afra 1111
cita 0000
```

if the sort command is run, the content of the file is ordered alphabetically:

```
professor@professor-VirtualBox:~$ sort file0.txt
afra 1111
bint 2222
cita 0000
lilkington 6789
lilypink 3456
madmax 4567
nolives 5678
redsell 2345
rose 1234
sutton 4444
```

In this case, the options command indicates numerically sorter and the field 2, that is to say, second column. The default delimiter is the space.

```
professor@professor-VirtualBox:~$ sort -nk2 file0.txt
cita 0000
afra 1111
rose 1234
bint 2222
redsell 2345
lilypink 3456
sutton 4444
madmax 4567
nolives 5678
lilkington 6789
```

In that example, options indicate to sort numerically, reverse direction and filed 2. The default delimiter is the space.

```
professor@professor-VirtualBox:~$ sort -nrk2 file0.txt
lilkington 6789
nolives 5678
madmax 4567
sutton 4444
lilypink 3456
redsell 2345
bint 2222
rose 1234
afra 1111
cita 0000
```

In the following case, the content file is the same but it is supposed that the delimiter is “.”.

```
professor@professor-VirtualBox:~$ cat file0.txt
rose,1234
redsell,2345
lilypink,3456
madmax,4567
nolives,5678
lilkington,6789
sutton,4444
bint,2222
afra,1111
cita,0000
```

```
professor@professor-VirtualBox:~$ sort -t, -bk2 file0.txt
cita,0000
afra,1111
rose,1234
bint,2222
redsell,2345
lilypink,3456
sutton,4444
madmax,4567
nolives,5678
lilkington,6789
```

As a field delimiter, the command takes the coma character pointed out with option `-t`. The rest of the options indicate to sort numerically, and the second field.

In addition to that use of sort, that command is used in order to join text file using redirect and pipes. But that will be learned in the future.

Due to the plenty of possibilities of that command (sort), it is recommended to check the help: `sort --help`

- **ln** \Rightarrow that command is used in order to create file or directory links. If the argument is a directory, the created link will be a symbolic link. By default, hard link is created. Options:

- `-s` \Rightarrow a symbolic link is created instead of hard link.

Example:

In order to create a hard link to the file `pepe.txt`:

1. `touch pepe.txt` \Rightarrow command touch creates an empty file
2. `ls -li pepe.txt` \Rightarrow check the i-node of pepe.txt
3. `ln pepe.txt hl_pepe` \Rightarrow to create a hard link of pepe.txt
4. Check if the file and hard link of the i-nodes are the same. Remember that the file and its hard-link share the same i-node.

Try it 6

- | Create a symbolic link of the file `pepe.txt`:

- which \Rightarrow it shows the command for absolute path of the arguments.
- locate \Rightarrow it searches files in the file system. The searches are through the file name.
- cut \Rightarrow it shows certain lines of the file of the argument.

Options:

- -b \Rightarrow it shows the specified bytes
- -c \Rightarrow it shows the specified characters
- -d \Rightarrow this option is used in order to communicate what character is going to be used as a delimiter.
- -f \Rightarrow it is used to indicate what fields should be shown

Examples: check out the examples in the following file:

```
professor@professor-VirtualBox:~$ cat file1.txt
arbol,casa,bosque,ciudad
avion,coche,camion,autopista
ordenador,pantalla,tablet,smartphone
```

As it is seen, the file is formed by 3 lines and 4 words per line. The delimiter is “,”. The first character of each line is shown with the command and options showed at the following image:

```
professor@professor-VirtualBox:~$ cut -c1 file1.txt
a
a
o
```

If it wanted to show the first 5 characters of each row, a range can be specified:

```
professor@professor-VirtualBox:~$ cut -c1-5 file1.txt
arbol
avion
orden
```

If the “,” delimiter is used, it is possible to cut columns, for example, to cut the last 2 columns:

```
professor@professor-VirtualBox:~$ cut -d"," -f3,4 file1.txt
bosque,ciudad
camion,autopista
tablet,smartphone
```

Pipes

Pipes are used to connect the standard output of some command with the standard input of another command. To do this, the character | is used. Example: Show the i-node of the files of your personal directory. Also show the type of file, permissions, and owner name of the files.

```

professor@professor-VirtualBox:~$ ls -li|cut -d " " -f1,2,4
total 48
1205202 drwxr-xr-x professor
1205206 drwxr-xr-x professor
1205203 drwxr-xr-x professor
1202355 -rw-r--r-- professor
1205223 -rw-r--r-- professor
1205207 drwxr-xr-x professor
1205208 drwxr-xr-x professor
1205205 drwxr-xr-x professor
1205204 drwxr-xr-x professor
271741 2

```

In the example, the output of the first command (`ls -li`) is used as an input for the second command (`cut -d " " -f1,2,3`). That command cuts the fields 1, 2 and 4; and specify as a delimiter the space “ ”.

- **tr** \Rightarrow translate, reduce and/or erase characters of the standard input, just writing in the standard output.

Syntax:

```
tr [option] [set1] [set2]
```

Options:

- **-c** \Rightarrow the complement of the set1 is used
- **-d** \Rightarrow the characters or group of them of the set1 are erased.
- **-s** \Rightarrow erase the repetition of the set1 character. Reduce to one character.
- **-t** \Rightarrow translate the coincidences of set1 to set2.

Examples: For the following examples, the output of the `ls -l` command will be used.

```

professor@professor-VirtualBox:~$ ls -l
total 48
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Desktop
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Documents
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Downloads
-rw-r--r-- 1 professor professor 8980 oct 25 17:32 examples.desktop
-rw-r--r-- 1 professor professor  91 oct 25 17:47 file1.txt
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Music
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Pictures
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Public
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Templates
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Videos

```

In order to reduce the repetitions of the “s” character, for example in professor:


```
professor@professor-VirtualBox:~$ ls -l | tr -s "s"
total 48
drwxr-xr-x 2 profesor profesor 4096 oct 25 17:42 Desktop
drwxr-xr-x 2 profesor profesor 4096 oct 25 17:42 Documents
drwxr-xr-x 2 profesor profesor 4096 oct 25 17:42 Downloads
-rw-r--r-- 1 profesor profesor 8980 oct 25 17:32 examples.desktop
-rw-r--r-- 1 profesor profesor  91 oct 25 17:47 file1.txt
drwxr-xr-x 2 profesor profesor 4096 oct 25 17:42 Music
drwxr-xr-x 2 profesor profesor 4096 oct 25 17:42 Pictures
drwxr-xr-x 2 profesor profesor 4096 oct 25 17:42 Public
drwxr-xr-x 2 profesor profesor 4096 oct 25 17:42 Templates
drwxr-xr-x 2 profesor profesor 4096 oct 25 17:42 Videos
```

The same with the repetition of the space:

```
professor@professor-VirtualBox:~$ ls -l | tr -s " "
total 48
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Desktop
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Documents
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Downloads
-rw-r--r-- 1 professor professor 8980 oct 25 17:32 examples.desktop
-rw-r--r-- 1 professor professor 91 oct 25 17:47 file1.txt
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Music
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Pictures
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Public
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Templates
drwxr-xr-x 2 professor professor 4096 oct 25 17:42 Videos
```

If it wanted to change the space delimiter between fields for “#” character:

```
professor@professor-VirtualBox:~$ ls -l | tr -t " " "#"
total#48
drwxr-xr-x#2#professor#professor#4096#oct#25#17:42#Desktop
drwxr-xr-x#2#professor#professor#4096#oct#25#17:42#Documents
drwxr-xr-x#2#professor#professor#4096#oct#25#17:42#Downloads
-rw-r--r--#1#professor#professor#8980#oct#25#17:32#examples.desktop
-rw-r--r--#1#professor#professor###91#oct#25#17:47#file1.txt
drwxr-xr-x#2#professor#professor#4096#oct#25#17:42#Music
drwxr-xr-x#2#professor#professor#4096#oct#25#17:42#Pictures
drwxr-xr-x#2#professor#professor#4096#oct#25#17:42#Public
drwxr-xr-x#2#professor#professor#4096#oct#25#17:42#Templates
drwxr-xr-x#2#professor#professor#4096#oct#25#17:42#Videos
```

4.3 Regular expressions

The regular expressions give an easy way to find a text string and defining character patterns. For instance, in order to find a group of words “cosa”, “casa”, “cesa”, the pattern “*c(o|a|e)sa*” would match the words.

To build a regular expression, some special characters are used in order to transmit what text string is being searched. Depending on what it wants to search, the following characters are used:

- Depending the position of the word:

Regular expression	Description
<code>^</code>	define a pattern that describes the start of the line using the special <code>^</code> (hat) metacharacter
<code>\$</code>	define a pattern that describes the end of the line using the special <code>\$</code> (dollar sign) metacharacter

For example:

- The lines which are started by “Linux” \Rightarrow “`^Linux`”
- The lines where the last character is “x” \Rightarrow “`x$`”
- The lines within the word “null” \Rightarrow “`^null$`”

Try it 7

- | The empty lines with no characters between the beginning and the end of the lines.

- A group of characters can be used in order to make the regular expression more powerful:

Regular expression	Description
<code>[abc]</code> <code>[a-z]</code>	In order to indicate any independent character, the first example could be used. In that case, the coincidences will be in the characters a, b or c It is possible to indicate a range of characters such as in the second example.
<code>[^abc]</code> <code>[^a-z]</code>	The special character <code>^</code> specifies the inverse coincidence. So, in that case, the coincidences will be any independent character which is not in a, b, or c or in the a-z range.
<code>\<Word\></code>	Word coincidences
<code>.</code>	Any character just one
<code>\</code>	This character is necessary in order to specify any special character. For example, <code>\$</code> . That character is used in order to specify end of line in regular expressions. So, if <code>\$</code> is needed to use as a character in a pattern, it is possible to used it with <code>\$</code> .

For example:

- Coincidences with “Linux” , “linux” \Rightarrow `[Ll]inux`
- Coincidences with 3 numerical digit consecutives: \Rightarrow `[0-9][0-9][0-9]`

Try it 8

- | – Coincidences with each row which starts by any non-numerical character.
- | – Coincidences with lines with 5 characters in a line.

- Modifiers

Regular expression	Description
*	In order to express an indeterminate number (0 or more) of the character or expression that precedes it
?	It expresses the preceding pattern or character, it is optional. For example: add? It would find ad and add It can be used with grep when -E option
+	It is used in order to find a string of characters which is repeated 1 or more times. For example: "[a-zA-Z][0-9]+" ⇒ H1 but not just H without any number It can be used with grep when -E option
\{n,m\}	It indicates a range of coincidences. \{n\} ⇒ specify at least n coincidences \{n,m\} ⇒ specify at least n and as maximum m coincidences. \{,m\} ⇒ No minimum, just maximum number of coincidences.
	It is used in order to indicate several options. For example: a e ⇒ a or e will be found It can be used with grep when -E option

For example:

- Coincidences with “ab”, “abc”, “abcc”, “abccc” ⇒ abc*
- Coincidences with “abc”, “abcc”, “abccc” ⇒ abc+
- Coincidences with 2 or more numerical digits. ⇒ [0-9]\{2,\}

Try it 9

- Coincidences with “fichero”, “fichero1”, “fichero2”.
- Coincidences with at least 1 numerical digit.
- Coincidences with “111”, “1111”, “11111”.
- Coincidences with “Happy”, “happy”, “Sad”, “sad”, “Angry” o “angry”.

4.4 grep command

It shows file lines which are coincided with a specified pattern Options:

- -r ⇒ it is used in order to find in recursive way inside of files of a directory.
- -n ⇒ show the number of the line of the coincidence
- -i ⇒ not sensitive to uppercase or lowercase
- -v ⇒ the lines which inverse coincidences
- -w ⇒ match whole word
- -c ⇒ this prints only a count of the lines that match a pattern

- -l⇒ displays list of “a” filenames only.

For example:

In order to develop these examples, they are used in the following filexmp1e.txt:

```
professor@professor-VirtualBox:~$ cat filexmp1e.txt
1
Linux
11
tree
house
1111
11111
linux
a
ab
abc
abcc
abccc
```

To find coincidences with Linux:

```
professor@professor-VirtualBox:~$ grep -En linux filexmp1e.txt
8:linux
```

To find Coincidences with “Linux” ,“linux” :

```
professor@professor-VirtualBox:~$ grep -En [lL]inux filexmp1e.txt
2:Linux
8:linux
```

To find coincidences with “abc”, “abcc”,“abccc”:

```
professor@professor-VirtualBox:~$ grep -En abc+ filexmp1e.txt
11:abc
12:abcc
13:abccc
```

To find coincidences with “ab”, “abc”, “abcc”,“abccc”:

```
professor@professor-VirtualBox:~$ grep -En abc* filexmp1e.txt
10:ab
11:abc
12:abcc
13:abccc
```

4.5 find command

It is a command in order to find files which coincide with the search expression from the pointed-out directories

Syntax:

```
find path options
```

Options:

- `-name` \Rightarrow it finds files whose name coincides with the given name. With `-iname` option, any difference will be made between uppercase and lowercase. It is possible to write several names and uses `-a` (and logical operation) `-o` (or logical operation) and `-not` (not logical operation)
- `-type` \Rightarrow it specifies the kind of file (f regular file, d directory and l symbolic link)
- `-maxdepth n` \Rightarrow it is used in order to specify maximum level of subdirectories where the search will be made.
- size:
 - `C` \Rightarrow byte, `k` \Rightarrow kilobyte
 - `+` \Rightarrow greater than
 - `-` \Rightarrow less than
 - No sign \Rightarrow it specifies the exact size
- `Inum n` \Rightarrow it searches the file i-node n.
- `perm` \Rightarrow it specifies the permissions
 - Octal
 - Symbolic permissions \Rightarrow `ugo+-rwx` (`-` specifies at least this permission, without `-` the exact permissions)
- `-links` \Rightarrow number of links
- `-user` \Rightarrow it specifies the owner of the file
- `-mtime` \Rightarrow it specifies the modified files in the last n days
- `-atime` \Rightarrow it specifies the accessed files in the last n days
- `-mmin` \Rightarrow it specifies the modified files in the last n minutes.
 - `+` \Rightarrow greater than
 - `-` \Rightarrow less than
 - No sign \Rightarrow it specifies the exact time

Actions \Rightarrow it is possible to run some action over the found files. The position of the file is pointed out with `{}` and the command ends with `\;`

Example: Search in the work directory all the files with doc or txt extension and move them to `/home/usuario/Documentos`

```
find -name "*.doc" -o -name "*.txt" -type f -exec mv {} /home/usuario/Documentos \;
```

Search every regular file greater than 512 kilobytes, and list them with the greater unit of measure:

```
find /home -type f -size +512k -exec ls -lh {} \;
```



Try it 10

Find the files with name file and every number (one numerical digit), for example: file0, file1, file2, file3 less than 256 bytes, and which owner is professor and list in large format.