When you work in the terminal, you may want to see the contents of the folders. It also helps to know the general information about the files: who created them, when, and so on. There is a separate utility for it in Unix systems called `ls`. Let's figure out how to use it and what are its main options.

Basic options

For starters, we should mention the command syntax. It looks like this:

`$ ls <options> /path/to/folder`.

Let's take the *test* folder as an example. In the simplest case, you can open it in a terminal and just write `ls` without any additional parameters. Then you will see all the files that are in this folder:

```
$ ls
```

```
my_python.py  Poems.pdf  script.sh  story.txt
```

Here you see only a list of file names. If you need to know more about these files, you will need the

`-l` option: the output will be a detailed list, which will display the owner of the file, its group, creation date, and size:

```
$ ls -l
total 52
-rw-r--r-- 1 alina alina  2578 feb 25 10:44 my_python.py
-r-------- 1 alina alina 26346 feb 18 22:21 Poems.pdf
-rw-r--r-- 1 alina alina    35 jan 28 11:30 script.sh
-rw-rw-r-- 1 alina alina 13405 jan 22 21:58 story.txt
```

So now you know who created the files, when they were created, and who has the rights to view, edit, or execute them. But there is a small problem

with the output: the actual size of the files looks messy. To make it easy to read we can combine the detailed view with

`-h` or `--human-readable` flag:

```
$ ls -lh
total 52
-rw-r--r-- 1 alina alina  2.6K feb 25 10:44 my_python.py
-r-------- 1 alina alina   26K feb 18 22:21 Poems.pdf
-rw-r--r-- 1 alina alina    35 jan 28 11:30 script.sh

-rw-rw-r-- 1 alina alina   14K jan 22 21:58 story.txt
```

This way we can see the size of the files rounded to the kibibytes (K), or, if they are big enough, rounded to mebibytes (M) or even gibibytes (G).

The output of the `ls` command by default only has the regular visible files. What if you need to view all the files including the hidden ones? Well, there is a separate `-a` parameter for this purpose:

```
$ ls -a
 .                                  .python_history
 ..                                 snap
 .android                           .sqldeveloper
 .audacity-data                     .ssh

 .bash_history                      .steam
```

Use all these options mentioned above to explore files in the given folder and find out information about them. Sometimes you may also want to sort the output in a particular order, so let's see how it's done in the next section.

Sorting files

First things first, you can sort the files by their size: either from largest to smallest, or vice versa. In this case, you need the

`-S`

parameter, and `-Sr` for the reverse order:

```
$ ls -sS
total 52
28 Poems.pdf  16 story.txt   4 my_python.py   4 script.sh

$ ls -sSr
total 52
 4 script.sh   4 my_python.py  16 story.txt  28 Poems.pdf
```

We've also included the `-s` flag to show the actual size of files, but it's not necessary for sorting.

The other option is to sort files by creation time. Use `-t` to sort files from recently created ones to those that were created a long time ago and use

`-tr` if you want to start from the oldest files:

```
$ ls -t
my_python.py  Poems.pdf  script.sh  story.txt

$ ls -tr
story.txt  script.sh  Poems.pdf  my_python.py
```

There are other ways to sort files, and you can read about them by running either `man ls` or `ls --help`.

In this section we've looked at the most basic options of the

`ls` command that allow you to display the files from a folder. What if we want to see the entire structure of our directories? The `tree` command will help us with that.

Tree catalog structure

The tree command displays the directory structure itself as a tree. To start using it, you need to install it with the commands

`sudo apt install tree` or `sudo snap install tree`

(if you have other package managers in your distro, the syntax for the installation command should be almost the same). Then you can go to the desired folder and see its structure. For example, let's take the same *test* folder. It has a fairly simple structure, it contains 4 files and doesn't have any subfolders:

```
$ tree
.
├── my_python.py
├── Poems.pdf
├── script.sh
└── story.txt

0 directories, 4 files
```

The `tree` command has additional options. For example, if you only want to see a list of all the directories, you can use the `-d` parameter. In our example, there are no subfolders, so let's create one to make it more illustrative. We've added a *literature* folder and moved the *Poems.pdf* and *story.txt* files there, so the `test`

folder has one directory inside of it:

```
$ tree -d
.
└── literature

1 directory
```

You can go further down the tree and choose the number of levels to observe. Use the parameter

`L` to walk down the levels. For example, let's go down two levels, so it's

`-L 2`. This way we can see not only the contents of the *test* folder but also the contents of the *literature* folder:

```
$ tree -L 2
.
├── literature
│   ├── Poems.pdf
│   └── story.txt
├── my_python.py
└── script.sh


1 directory, 4 files
```

Great, now you can view your folder structures! However, we should take one more point into account. There are different types of files in the directories, so it would be useful to know how to quickly and easily examine their purpose. There is a special command for it, which is called `file`.

File content

The `file` command lets you know the type of data that is actually contained within a document. To use it, type in the terminal `file document_name`. For example, let's take a closer look at the *story.txt* file in the *literature* folder:

```
$ file story.txt

story.txt: UTF-8 Unicode text, with very long lines
```

As we can see, the file contains text in UTF-8 encoding. For the next example, let's take the *script.sh* executable file:

```
$ file script.sh

script.sh: Bourne-Again shell script, ASCII text executable
```

Now you can recognize the contents of a file without opening it.

Conclusion

So, in this topic we've figured out how to view the contents of the folders and sort files using the `ls` command and how to visualize the directory structure using the `tree` command. We've also learned how to find out about the contents of a file without opening it using the `file` command.

Exploring details of all files with less

Consider that you are in a directory with multiple folders and files. You want to list all the files of a specific folder including hidden files, output them in a long listing format and pipe this output into the `less` command for easy viewing. Which of the following commands will help you achieve this?

```
ls folder_name -al | less
```

What will you see on the screen if you type the `tree -d -L 3` command?
Structure of the directories 3 levels deep from the current folder

Command capabilities

Complete the command below to view the contents of a folder named `my_folder` in a format that includes permissions, owner, file size, and modification time, all sorted by modification time. Fill in the missing parts to complete the command.

Ls -tl my_folder

Which command should you use to find out the type of content in the file without opening it? File

Write a program that will print all the files including the hidden ones in the `/tmp/test` directory.

```
.
..
.hidden
.ignore
Poems.pdf
my_python.py
```

```
#!/usr/bin/env bash
solve() {
   ls -a /tmp/test

}
```

Sort files by size

Report a typo

Write a program that will sort the files from the

`/tmp/test`

folder by size in descending order and display the files and their size in the terminal.

Expected format:

```
#!/usr/bin/env bash
solve() {
   # add your solution here

}
```

```
#!/usr/bin/env bash
solve() {
   ls -sS /tmp/test}
```