Linux Server Administration: Introductory Course

Víctor Granda

Contents

Preface

In this introductory course we will explore the basics about accesing, using and maintaining a Linux server. Among other things, we will cover:

- Accessing remote servers with SSH
- Most common system commands and terminal use
- Users administration and permissions
- Transfering data between remote servers and local machines

1. Introduction

One or more remote servers can quickly become an useful resource to any research team. Depending on the server, they offer computational resources bigger than those available in our local computers. They come in handy to host tools and applications, reachable for the whole team, and they can also host documentation, databases, web pages for projects or groups...

> **Note**
>
> For example, at the EMF we have several servers, hosting the EMF web, the Forestry Lab, a notification server for automatized alerts, a documentation web, a file server, an internal resources database...

But, *with great power comes great responsability*. Remote servers must be maintained, users must be added and removed, applications must be installed, updated or removed and web servers must be served...
The next chapters will guide you step by step to use and maintain you remote linux server.

2. Accessing the server

The first thing we need to work with a remote server is to access to it. We are going to use the ***S**ecure **SH**ell* (SSH) protocol for this. SSH is a protocol to connect two computers over a network. The secure part means that contents will be encrypted on transport, so no plain text information will be send (important to avoid password and keys leaking). Basically this means that we will be using a terminal as if we were sitting in front of the server.

We also need two other things, an **address** and a **port**.

- To be able to reach the server, we need to know *where it is*. This is usually a numeric IP address unless a domain is set for the server.
- Besides where the server is, we need to know which *port* in the server we have to connect to. Think of ports as doors to the server, each of them specific for a protocol. When connecting to a website, we connect usually to the port **80**. When we are using SSH the default port is **22** unless is changed by the server administrator

> ! Important
>
> In this course we are going to practice with a server located at **144.76.203.10** and we are going to use the port **2222**

2.1 Connecting to the server

2.1.1 Opening a terminal in our computer

To connect to the server with SSH we need a terminal. In Windows (10 and 11) we can use the Windows Terminal or PowerShell (Figure 2.1, Figure 2.2). Mac users can use the Terminal or the iTerm2 apps. Linux users can use their preferred terminal and shell (Figure 2.3).

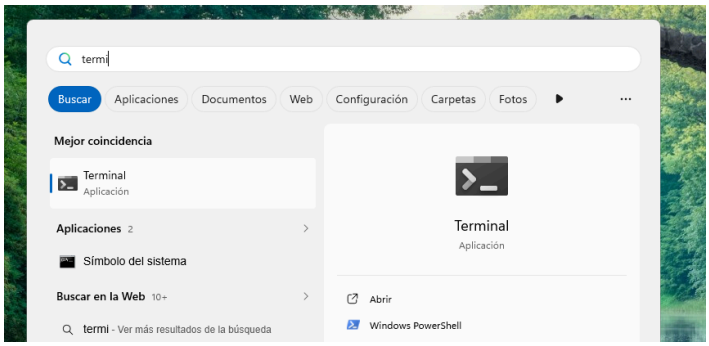

Figure 2.1: Launching terminal on Win11

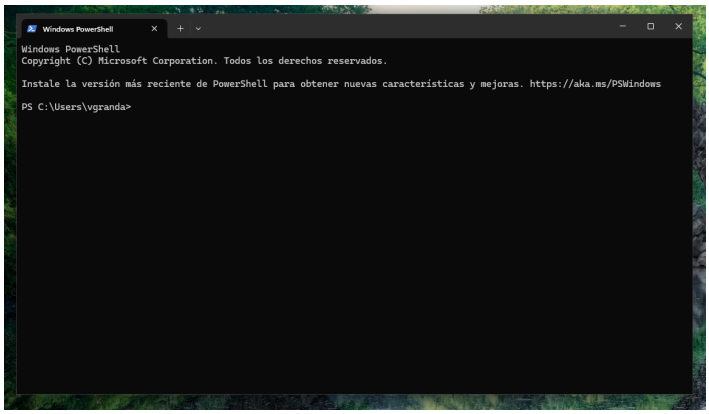

Figure 2.2: Terminal on Win11

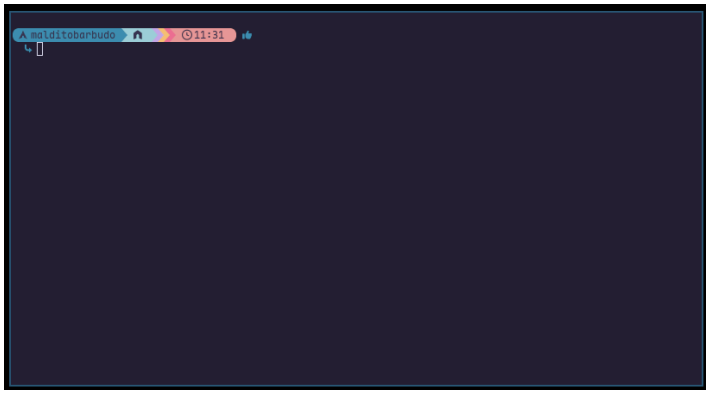

Figure 2.3: Linux terminal (ghostty)

2.1.2 `ssh` command

Once we have the terminal open, we just need to execute the `ssh` command, indicating the user and the address (also the port if is not the standard **22**)

```
ssh -p 2222 victor@144.76.203.10
```

Let's dissect the command:

- `ssh`: SSH command
- `-p 2222`: the port we need to connect to is specified with the `-p` flag
- `victor@144.76.203.10` user and address, separated by an @, you should change this with your username

If we execute (press Enter) this command, we will be prompted for the user's password.

> **Caution**
>
> As you enter the password, you will notice no characters appears in the screen. This is normal. Just type your password and press Enter.

2.1.3 First connection

The first time we connect to a server, a warning will appear after entering the password:

```
The authenticity of host '[144.76.203.10]:2222 ([144.76.203.10]:2222)' can't be
established.
ED25519 key fingerprint is: SHA256:****************************************
```

```
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

This is to ensure we are connecting to the correct server. Type `yes` and press Enter to add the server to the list of known hosts.

2.1.4 Using the server

Now we are *in* the remote server. We have several welcome messages:

```
Welcome to Ubuntu 24.04.4 LTS (GNU/Linux 6.8.0-124-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Jun 10 12:27:01 PM CEST 2026

  System load:  0.0               Processes:               170
  Usage of /:   0.2% of 1.77TB    Users logged in:         0
  Memory usage: 1%                IPv4 address for enp2s0: 144.76.203.10
  Swap usage:   0%                IPv6 address for enp2s0: 2a01:4f8:200:8409::2
  Temperature:  44.0 C

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

10 updates can be applied immediately.
7 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Wed Jun 10 10:59:19 2026 from 83.48.65.148
```

Also, the prompt has changed

```
victor@aula:~$
```

- `victor` indicates the user connected
- `aula` is the server name
- `~` is the folder we are. `~` is a shorthand for the user personal folder
- `$` indicates that we are a regular user

In the following chapters we will explore the commands and tools needed to interact with the server.

3. Managing users and permissions

As administrators of a linux server, we often need to perform changes in configurations, install software or create users accounts for our collegues or guests. This can't be done without some privileges, as a normal user can only modify its own personal folder, while the rest of the system is read-only.

3.1 `root` user

Every Linux installation has a `root` user account. This `root` has all privileges and can access, create, modify, remove files from any folder, included system folders. Also has access to system processes and the kernel.

> **i** Note
>
> This is why using the `root` account directly is **discouraged**, as it is a risk. We can, however, grant some privileges to some users to be able to act as `root` when needed.

3.2 `sudo` access

`sudo` is the tool we are going to use to act as administrators (**s**uper**u**sers). For example, if we want to check if there is any system update, we need to run `apt update`. Let's see what happens when we run it as a normal user:

```
victor@aula:~$ apt update
Reading package lists... Done
E: Could not open lock file /var/lib/apt/lists/lock - open (13: Permission denied)
E: Unable to lock directory /var/lib/apt/lists/
W: Problem unlinking the file /var/cache/apt/pkgcache.bin - RemoveCaches (13:
Permission denied)
W: Problem unlinking the file /var/cache/apt/srcpkgcache.bin - RemoveCaches (13:
Permission denied)
```

Now, let's see what happens when run as a superuser:

```
victor@aula:~$ sudo apt update
[sudo] password for victor:
Hit:1 http://mirror.hetzner.com/ubuntu/packages noble InRelease
Hit:2               http://mirror.hetzner.com/ubuntu/packages                noble-
updates InRelease
Hit:3               http://mirror.hetzner.com/ubuntu/packages                noble-
backports InRelease
Hit:4               http://mirror.hetzner.com/ubuntu/packages                noble-
security InRelease
Hit:5                   http://security.ubuntu.com/ubuntu                    noble-
security InRelease
Hit:6 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:7 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:8 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
5 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

But, when run by an user that don't have `sudo` privileges, the command fails:

```
test_user@aula:~$ sudo apt update
[sudo] password for test_user:
test_user is not in the sudoers file.
```

This means that `test_user` can't modify or do anything harmful in the system, only access its own home folder and little else.

> Tip
>
> Is always recommended to create users without `sudo` access, except for the person responsible of the server, the administrator. `sudo` access can always be granted later in case it's needed.

3.3 Creating users

To create new users, we are going to use the `useradd` command. We are going to need serveral things:

- An username
- An user folder at `/home/username` (flag `-m`)
- A shell, in this case `/usr/bin/bash` (flag `-s`)
- (optional) A group like `sudo` in case the user is going to have administrative privileges (flag `-G`)
- A password for the user, we will add it after the user creation with `passwd`

Let's see an example:

```
victor@aula:~$ sudo useradd -m -s /usr/bin/bash -G sudo test_user_2
victor@aula:~$ sudo passwd test_user_2
New password:
Retype new password:
passwd: password updated successfully
```

We have succesfully created an user with administrative privileges. We can check it by login into the server with the newly created user and try to check for system updates:

```
test_user_2@aula:~$ sudo apt update
[sudo] password for test_user_2:
Hit:1 http://mirror.hetzner.com/ubuntu/packages noble InRelease
Hit:2                http://mirror.hetzner.com/ubuntu/packages                noble-
updates InRelease
Hit:3                http://mirror.hetzner.com/ubuntu/packages                noble-
backports InRelease
Hit:4                http://mirror.hetzner.com/ubuntu/packages                noble-
security InRelease
Hit:5                                                  http://archive.ubuntu.com/ubuntu
noble InRelease
Hit:6 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:8 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
5 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

In case we need to modify something, we can use the `usermod` command. For example, to grant administrative privileges to the `user_test` user, we can do:

```
victor@aula:~$ sudo usermod -aG sudo test_user
[sudo] password for victor:
```

Here, the flags `-aG sudo` means add to sudo Group.

3.4 Exercise

> **i** Exercise: Lets create users for everyone!
>
> In this exercise you are going to work by pairs (if possible, by 3 people if not).
> Using the `test_user` account (password `test_user`) you are going to create an user for your partner and give it a password. Then, you will have to check that everything worked as intended and both/all of you can access the server with your brand new aacounts.
>
> Remember, use `useradd` with the necessary flags for creating the user, and don't forget to add the password with `passwd` or the new user will not be allowed to access the server.

3.5 Deleting users

In order to maintain a healthy remote server, we also need to periodically remove old users that are not working anymore in the server. This can be done with `userdel`. For example, now that we don't need anymore the `test_user` and `test_user_2` users, we can safely remove them from the system, as well as their personal folders:

```
victor@aula:~$ sudo userdel -r test_user_2
[sudo] password for victor:
userdel: test_user_2 mail spool (/var/mail/test_user_2) not found
victor@aula:~$ sudo userdel -r test_user
userdel: test_user mail spool (/var/mail/test_user) not found
```

4. Navigation

In this chapter we will learn the basics of the linux folder structure and how to create folders and files and navigate it.

4.1 Linux folders structure (Filesystem Hierarchy Standard)

Linux filesystem follows the FHS (https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard). This hierarchy defines the names and location of the system folders, containing the necessary files and configurations to run the operative system.
In Linux, the base of the filesystem is `/`, and it serves a the `root folder` where the rest of the hierarchy resides. The most relevant folders for our course are:

- `/home`: Users' personal folders
- `/bin`: Commands and tools (binaries) that are necessary to run and maintain the system.
- `/etc`: System wide configuration files.

> ⚠ Warning
>
> There is an important difference here between Windows and Linux, and is the character used to separate folders when describing a path to a file. In Linux it would be
> `/home/victor/Documents/my_file.txt`
> while in Windows is
> `C:\Users\victor\Documents\my_file.txt`

4.2 Listing files

When you login in a linux server you will be at your user personal folder. It can be checked with the `pwd` (**p**rint **w**orking **d**irectory) command:

```
victor@aula:~$ pwd
/home/victor
```

In this case, I'm in the folder `victor` (my personal folder), located at `/home`. To see what's inside of the folder, we can use the `ls` (**ls**t) command:

```
victor@aula:~$ ls
victor@aula:~$
```

As my user is new, and I haven't created any file or subfolder yet, my personal folder is empty. **But is it really?**

```
victor@aula:~$ ls -a
.     ..     .bash_history     .bash_logout     .bashrc     .cache     .cloud-locale-
test.skip  .profile  .sudo_as_admin_successful
```

> 💡 Tip
>
> Most linux commands and tools have a manual that can tell us the options (flags) we have available. In this case you can use `man ls` to see all the possibilities.

> **i** Note
>
> Files and folders in Linux starting with a dot (`.`) are hidden, and they don't appear unless we use the `-a` flag.

4.3 File operations (create, copy, move, delete)

As a normal user, we can access most of the folders (`/bin`, `/etc`...), but we can only modify our personal folder.

> **i** Note
>
> We will see later how to become a *superuser* with permissions to modify system configurations and installing new tools and software.

4.3.1 Creating folders

We can create folders with `mkdir` (**m**a**k**e **dir**ectory). For example, if we are in our home folder and we want to create a folder named `projects` we do the following:

```
victor@aula:~$ mkdir projects
```

And now if we check with `ls` we can see the new folder:

```
victor@aula:~$ ls -la
total 36
drwxr-x--- 4 victor victor 4096 Jun 24 10:09 .
drwxr-xr-x 4 root   root   4096 Jun  2 15:17 ..
-rw------- 1 victor victor  164 Jun 11 12:55 .bash_history
-rw-r--r-- 1 victor victor  220 Jun  2 15:17 .bash_logout
-rw-r--r-- 1 victor victor 3771 Jun  2 15:17 .bashrc
drwx------ 2 victor victor 4096 Jun  2 15:21 .cache
-rw-r--r-- 1 victor victor    0 Jun  2 15:17 .cloud-locale-test.skip
-rw------- 1 victor victor   20 Jun 24 09:22 .lesshst
-rw-r--r-- 1 victor victor  807 Jun  2 15:17 .profile
drwxrwxr-x 2 victor victor 4096 Jun 24 10:09 projects
-rw-r--r-- 1 victor victor    0 Jun  2 15:22 .sudo_as_admin_successful
```

If we want to create a hierarchy of folders, for example `projects/CREAF/linux_course`, we can do it using the `-p` flag. This flag tells `mkdir` to create any intermediate folder that don't exists yet:

```
victor@aula:~$ mkdir -p projects/CREAF/linux_course
victor@aula:~$ ls -la projects/CREAF/
total 12
drwxrwxr-x 3 victor victor 4096 Jun 24 10:16 .
drwxrwxr-x 3 victor victor 4096 Jun 24 10:16 ..
drwxrwxr-x 2 victor victor 4096 Jun 24 10:16 linux_course
```

4.4 Moving through folders

To move between folders we can use `cd` (**c**hange **d**irectory). So, to navigate to the recently created folder:

```
victor@aula:~$ cd projects/CREAF/linux_course
victor@aula:~/projects/CREAF/linux_course$
```

We can see that the prompt has changed, indicating now that we are in the `/home/victor/projects/CREAF`.

Tip

In linux we can use *relative paths* or *absolute paths*. A relative path takes as base our current working directory as the base and buiold the path from there. An absolute path always start in the root folder (`/`). So, if we are in `/home/victor` and want to move to `/home/victor/projects/CREAF/linux_course` these two are equivalent:

- absolute path: `victor@aula:~$ cd /home/victor/projects/CREAF/linux_course`
- relative path: `victor@aula:~$ cd projects/CREAF/linux_course`

4.5 Creating empty files

Sometimes we need to create empty files to start working. This can be easily done with `touch`:

```
victor@aula:~/projects/CREAF/linux_course$ touch analysis.R
victor@aula:~/projects/CREAF/linux_course$ ls -la
total 8
drwxrwxr-x 2 victor victor 4096 Jun 25 10:15 .
drwxrwxr-x 3 victor victor 4096 Jun 24 10:16 ..
-rw-rw-r-- 1 victor victor    0 Jun 25 10:15 analysis.R
```

4.6 Editing files

For quick and short edits, we can use `nano`, a simple terminal editor available in most linux servers:

```
victor@aula:~/projects/CREAF/linux_course$ nano analysis.R
```

4.7 Copying files

We can **c**o**p**y files with `cp`, providing the source and the destination:

```
victor@aula:~/projects/CREAF/linux_course$ cp analysis.R analysis_old.R
```

For copying folders, we need the `-r` flag to **r**ecursively copy all files and folders inside.

Caution

If the destination exists, it **WILL BE OVERWRITTEN** without any warning

If instead of copying we want to move the files from one folder to another, the we can use `mv` in the same way as we use `cp`.

4.8 Deleting files

To **re**m**o**ve (delete) files we can use `rm` and the path to the file (using the `-r` flag for folders).

Caution

This removes permanently the files or folders selected. There is no way back, so be sure what you really want to remove and double check the spelling.

5. Data

We saw in the previous chapter how to create and edit files. This can be useful in some cases, but most of the time we will need to transfer files or folders between our local computer and the server or the other way around.

5.1 Moving files

There are different tools that can help us transfer data between computers through `ssh`, the same protocol we use to connect to remote servers:

- `scp` (**s**ecure **c**o**p**y). This is a command line utility found in all Linux, Mac and Windows that creates an SSH connection and tranfer the files.
- FileZilla. This is a graphical user interface that allows transfering files through different protocolos, SSH included.

5.1.1 `scp`

`scp` is a command line tool. That means that we will use the terminal to execute it. We can simply call the program without any arguments or flags to see a quick help:

```
victor@aula:~$ scp
usage: scp [-346ABCOpqRrsTv] [-c cipher] [-D sftp_server_path] [-F ssh_config]
           [-i identity_file] [-J destination] [-l limit] [-o ssh_option]
           [-P port] [-S program] [-X sftp_option] source ... target
```

> Tip
>
> Remember, you can have a more detailed explanation of all the options by executing `man scp`.

Let's see how to use `scp`. In this example I'm going to copy a zip file from my Windows machine to the `projects/CREAF/linux_course` folder I created in the previous chapter.
If you want to replicate the example, change the file for one in your computer that you want to copy.
In this case we know that we need the port 2222, so we will have to use the `-P` flag to provide the port number. Also, we need a path to the source file in our computer and a path to the target (destination) file in the remote server.

```
PS C:\Users\vgranda> scp -P 2222 '.\Documents\ifn4_acoruna_tcm30-536596.zip'
victor@144.76.203.10:~/projects/CREAF/linux_course/ifn_coruña.zip
victor@144.76.203.10's password:
ifn4_acoruna_tcm30-536596.zip                                 100%
4076KB  83.9KB/s   00:48
```

And that's it. We can connect to the server and list the files in the `linux_course` folder to see if the file is really there:

```
PS C:\Users\vgranda> ssh -p 2222 victor@144.76.203.10
victor@144.76.203.10's password:
victor@aula:~$ ls -la projects/CREAF/linux_course/
total 4084
drwxrwxr-x 2 victor victor    4096 Jun 29 13:41 .
drwxrwxr-x 3 victor victor    4096 Jun 24 10:16 ..
-rw-rw-r-- 1 victor victor       0 Jun 25 10:15 analysis.R
-rw-rw-r-- 1 victor victor 4173392 Jun 29 13:42 ifn_coruña.zip
```

Tip

What happens when we want to copy a file from the remote server to our local computer? In this case we just simply indicate the source in the remote server and the target in our computer:

```
PS C:\Users\vgranda> scp -P 2222 victor@144.76.203.10:~/projects/CREAF/
linux_course/ifn_coruña.zip '.\Documents\ifn_coruña.zip'
```

To copy folders the process is the same, but we have to add the `-r` flag to make the copy recursive to all files and subfolders contained in the folder we want to copy:

```
PS C:\Users\vgranda> scp -P 2222 -r '.\Documents\project_iris' victor@144.76.203.10:~/
projects/CREAF/linux_course/
```

5.1.2 FileZilla

FileZilla (https://filezilla-project.org/) is a free and open source graphical interface to access and transfer files between computers. Is out of the scope of this course, but you can check their website to learn how to install it and use it.

5.2 Disk space

When transfering files to the server, is important to be aware of the available space in the disk. Not only the transfer will stop with an error if the disk is full, but some system process can be also affected as they can depend on having enopugh space to create temporal files.
There are two command line utilities that can help us to check the available space in disk, and also what folders and files are using that space

5.2.1 `df`

`df` is an utility to quickly check the used and available space in the disk:

```
victor@aula:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           3.2G  944K  3.2G   1% /run
/dev/md2        1.8T  2.9G  1.7T   1% /
tmpfs            16G     0   16G   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/md1        989M  202M  737M  22% /boot
tmpfs           3.2G   12K  3.2G   1% /run/user/1001
```

The `-h` flag is just simply converting the space to human readable units (GB, TB, MB...)

5.2.2 `du`

`du` is another utility that analyse the disk and tell us the size of folders and files:

```
victor@aula:~$ du -h -d 1 /home/victor
4.0K    /home/victor/.cache
4.2M    /home/victor/projects
4.2M    /home/victor
```

The `-h` flag is the same as before and the `-d 1` flag indicates to go only 1 level deep.

6. File permissions

Now that we have user accounts and administrative privileges we need to understand better differences between users and groups and also how file permissions works in linux.

Let's check with our users the contents of our personal folder with `ls` as we learn in the previous chapters, using the `-la` flags:

```
victor@aula:~$ ls -la
total 40
drwxr-x--- 5 victor victor 4096 Jun 29 15:30 .
drwxr-xr-x 5 root   root   4096 Jun 30 12:40 ..
-rw------- 1 victor victor 1608 Jun 30 12:41 .bash_history
-rw-r--r-- 1 victor victor  220 Jun  2 15:17 .bash_logout
-rw-r--r-- 1 victor victor 3771 Jun  2 15:17 .bashrc
drwx------ 2 victor victor 4096 Jun  2 15:21 .cache
-rw-r--r-- 1 victor victor    0 Jun  2 15:17 .cloud-locale-test.skip
-rw------- 1 victor victor   20 Jun 29 14:35 .lesshst
drwx------ 3 victor victor 4096 Jun 29 15:30 .local
-rw-r--r-- 1 victor victor  807 Jun  2 15:17 .profile
drwxrwxr-x 3 victor victor 4096 Jun 24 10:16 projects
-rw-r--r-- 1 victor victor    0 Jun  2 15:22 .sudo_as_admin_successful
```

Let's focus in one of the lines

```
drwxrwxr-x 3 victor victor 4096 Jun 24 10:16 projects
```

- `drwxrwxr-x` is the permissions description
- `victor victor` are the username and group of the file/folder owner
- `4096 Jun 24 10:16` are metadata (size, modify date...)

> How to read the permissions description
>
> First `d` indicates a folder.
> First `rwx` indicates permission to read, write and execute for the owner of the file.
> Second `rwx` block indicates permission to read, write and execute for members of the owner group.
> Third `r-x` block indicates permission only to read and execute for everyone else (users not in the owner group).

> Exercise: Describe permissions
>
> Describe the following file permissions:
>
> - `drwx------ 2 victor victor 4096 Jun  2 15:21 .cache`
> - `-rw-r--r-- 1 victor victor 3771 Jun  2 15:17 .bashrc`

6.1 How to change permissions and ownership

Permissions and ownership can be changed. This allow us to create folders that any user in the same group can write to and read from. For this we need to:

1. Create a group
2. Add your user to the new group
3. Create a folder
4. Change the default owner group to the newly created group

5. Add other users to the newly created group

6.1.1 Creating the group and the folder

`groupadd` allows us to create a group, and as we saw before, `usermod` can serve us to add our user to the group:

```
victor@aula:~$ sudo groupadd research
[sudo] password for victor:
victor@aula:~$ sudo usermod -aG research victor
```

We need to exit and connect again for the changes to take effect. After that we can create the folder:

```
victor@aula:~$ mkdir research
victor@aula:~$ ls -la
total 44
drwxr-x--- 6 victor victor 4096 Jun 30 13:24 .
drwxr-xr-x 5 root   root   4096 Jun 30 12:40 ..
-rw------- 1 victor victor 1716 Jun 30 13:20 .bash_history
-rw-r--r-- 1 victor victor  220 Jun  2 15:17 .bash_logout
-rw-r--r-- 1 victor victor 3771 Jun  2 15:17 .bashrc
drwx------ 2 victor victor 4096 Jun  2 15:21 .cache
-rw-r--r-- 1 victor victor    0 Jun  2 15:17 .cloud-locale-test.skip
-rw------- 1 victor victor   20 Jun 30 13:23 .lesshst
drwx------ 3 victor victor 4096 Jun 29 15:30 .local
-rw-r--r-- 1 victor victor  807 Jun  2 15:17 .profile
drwxrwxr-x 3 victor victor 4096 Jun 24 10:16 projects
drwxrwxr-x 2 victor victor 4096 Jun 30 13:24 research
-rw-r--r-- 1 victor victor    0 Jun  2 15:22 .sudo_as_admin_successful
```

6.1.2 Changing the ownership of the folder

As it is now, the research folder is only accessible by my user (`victor`) and by users belonging to the `victor` group. To change this we need to use `chown`:

```
victor@aula:~$ chown -R victor:research research
victor@aula:~$ ls -la
total 44
drwxr-x--- 6 victor victor   4096 Jun 30 13:24 .
drwxr-xr-x 5 root   root     4096 Jun 30 12:40 ..
-rw------- 1 victor victor   1716 Jun 30 13:20 .bash_history
-rw-r--r-- 1 victor victor    220 Jun  2 15:17 .bash_logout
-rw-r--r-- 1 victor victor   3771 Jun  2 15:17 .bashrc
drwx------ 2 victor victor   4096 Jun  2 15:21 .cache
-rw-r--r-- 1 victor victor      0 Jun  2 15:17 .cloud-locale-test.skip
-rw------- 1 victor victor     20 Jun 30 13:23 .lesshst
drwx------ 3 victor victor   4096 Jun 29 15:30 .local
-rw-r--r-- 1 victor victor    807 Jun  2 15:17 .profile
drwxrwxr-x 3 victor victor   4096 Jun 24 10:16 projects
drwxrwxr-x 2 victor research 4096 Jun 30 13:24 research
-rw-r--r-- 1 victor victor      0 Jun  2 15:22 .sudo_as_admin_successful
```

The `-R` flag makes the changes recursive to any file and subfolder contained in `research`.

Also, we maybe want that **only** the members in the `research` group can read or access the folder. For this we need to change the permissions with `chmod`:

```
victor@aula:~$ chmod -R u=rwx,g=rwx,o= research
victor@aula:~$ ls -la
total 44
drwxr-x--- 6 victor victor   4096 Jun 30 13:32 .
drwxr-xr-x 5 root   root     4096 Jun 30 12:40 ..
-rw------- 1 victor victor   1716 Jun 30 13:20 .bash_history
```

```
-rw-r--r-- 1 victor victor    220 Jun  2 15:17 .bash_logout
-rw-r--r-- 1 victor victor   3771 Jun  2 15:17 .bashrc
drwx------ 2 victor victor   4096 Jun  2 15:21 .cache
-rw-r--r-- 1 victor victor      0 Jun  2 15:17 .cloud-locale-test.skip
-rw------- 1 victor victor     20 Jun 30 13:32 .lesshst
drwx------ 3 victor victor   4096 Jun 29 15:30 .local
-rw-r--r-- 1 victor victor    807 Jun  2 15:17 .profile
drwxrwxr-x 3 victor victor   4096 Jun 24 10:16 projects
drwxrwx--- 2 victor research 4096 Jun 30 13:24 research
-rw-r--r-- 1 victor victor      0 Jun  2 15:22 .sudo_as_admin_successful
```

Now we have a folder, `/home/victor/research` that can be accessed only by members of the `research` group.

Tip

To ensure that any new file created in the folder inherits the correct group, we can use:

```
victor@aula:~$ sudo chmod g+s research
```

6.2 Exercise

Modifying permissions and ownership

Work in pairs (or groups of 3) to create a folder that only both/any of you can access, write to and read from.

7. Installing and updating software (system and apps)

Keeping the system updated is an important step in any server maintenance, not only Linux. An up-to-date system removes known security risks and provides new versions of the installed software.

> ! Important
>
> Maintaning the system updated is not failproof. A system can be compromised in many ways, some avoidable, others out of our hands. But there is some things we can do to minimize the impact of any security breach:
>
> 1. Always maintain a backup of the necessary data outside the server
> 2. Rotate credentials (passwords and keys) periodically
> 3. Delete old users

7.1 Software repositories

One difference between a Linux system and a Windows system is that in Linux, every distribution (linux flavours) have its own curated repository from which you can install software. This repository is officially maintained by the distribution and is the recommended place from which install any tool you need. Only in the case that the software you want is not in the official repositories that we will need to install it manually, generallyu foloowing the instructions on the software web page.

7.2 Updating the system

In Ubuntu (the linux distribution the server in this course uses), the tool to install and update the system is `apt`. As we've seen before, we need to have administrative privileges for being able to use it.

To check if there is updates available we use `apt update`, to install any available update, we use `apt upgrade`.

```
victor@aula:~$ sudo apt update
[sudo] password for victor:
victor@aula:~$ sudo apt upgrade
```

This will set the system up to the last versions of all installed software.

> Tip
>
> In other distributions the command is different from `apt`:
>
> - In ArchLinux is `pacman`
> - In Fedora or RHEL is `dfn`

7.3 Installing new software

Installing software present in the distribution repositories is straight forward. We use the `apt` command with the `install` options. For example, if we want to install R, we can do:

```
victor@aula:~$ sudo apt install r-base
Reading package lists... Done
Building dependency tree... Done
```

```
Reading state information... Done
The following packages were automatically installed and are no longer required:
    bpfcc-tools  bpftrace  ieee-data  libbpfcc  libclang-cpp18  libclang1-18  libllvm18
python3-bpfcc python3-netaddr
  ubuntu-kernel-accessories
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
   binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2 bzip2-doc
cpp cpp-13
 cpp-13-x86-64-linux-gnu cpp-x86-64-linux-gnu dpkg-dev fakeroot fontconfig fontconfig-
config fonts-dejavu-core
  fonts-dejavu-mono g++ g++-13 g++-13-x86-64-linux-gnu g++-x86-64-linux-gnu gcc gcc-13
gcc-13-base
    gcc-13-x86-64-linux-gnu  gcc-x86-64-linux-gnu  gfortran  gfortran-13  gfortran-13-
x86-64-linux-gnu
  gfortran-x86-64-linux-gnu icu-devtools libalgorithm-diff-perl libalgorithm-diff-xs-
perl libalgorithm-merge-perl
  libasan8 libatomic1 libauthen-sasl-perl libbinutils libblas-dev libblas3 libbz2-dev
libcairo2 libcc1-0
 libclone-perl libctf-nobfd0 libctf0 libdata-dump-perl libdatrie1 libdeflate0 libdpkg-
perl libdrm-amdgpu1
 libdrm-intel1 libegl-mesa0 libegl1 libencode-locale-perl libfakeroot libfile-basedir-
perl
    libfile-desktopentry-perl  libfile-fcntllock-perl  libfile-listing-perl  libfile-
mimeinfo-perl libfont-afm-perl
  libfontconfig1 libgbm1 libgcc-13-dev libgfortran-13-dev libgfortran5 libgl1 libgl1-
mesa-dri libgles2 libglvnd0
  libglx-mesa0 libglx0 libgomp1 libgprofng0 libgraphite2-3 libharfbuzz0b libhtml-form-
perl libhtml-format-perl
    libhtml-parser-perl  libhtml-tagset-perl  libhtml-tree-perl  libhttp-cookies-perl
libhttp-daemon-perl
   libhttp-date-perl  libhttp-message-perl  libhttp-negotiate-perl  libhwasan0  libice6
libicu-dev libio-html-perl
  libio-socket-ssl-perl libio-stringy-perl libipc-system-simple-perl libisl23 libitm1
libjbig0 libjpeg-dev
   libjpeg-turbo8  libjpeg-turbo8-dev  libjpeg8  libjpeg8-dev  liblapack-dev  liblapack3
liblerc4 libllvm20 liblsan0
    liblwp-mediatypes-perl  liblwp-protocol-https-perl  liblzma-dev  libmailtools-perl
libmpc3 libncurses-dev
      libnet-dbus-perl   libnet-http-perl   libnet-smtp-ssl-perl   libnet-ssleay-perl
libpango-1.0-0 libpangocairo-1.0-0
  libpangoft2-1.0-0 libpaper-utils libpaper1 libpciaccess0 libpcre2-16-0 libpcre2-32-0
libpcre2-dev
   libpcre2-posix3  libpixman-1-0  libpkgconf3  libpng-dev  libpng-tools  libquadmath0
libreadline-dev libsframe1
  libsharpyuv0 libsm6 libstdc++-13-dev libtcl8.6 libthai-data libthai0 libtie-ixhash-
perl libtiff6
  libtimedate-perl libtk8.6 libtry-tiny-perl libtsan2 libubsan1 liburi-perl libvulkan1
libwayland-client0 libwebp7
  libwww-perl libwww-robotrules-perl libx11-6 libx11-data libx11-protocol-perl libx11-
xcb1 libxau6 libxaw7
  libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-render0 libxcb-shape0
libxcb-shm0 libxcb-sync1
   libxcb-xfixes0  libxcb1  libxcomposite1  libxcursor1  libxdmcp6  libxext6  libxfixes3
libxft2 libxi6 libxinerama1
   libxkbfile1  libxml-parser-perl  libxml-twig-perl  libxml-xpathengine-perl  libxmu6
libxmuu1 libxpm4 libxrandr2
  libxrender1 libxshmfence1 libxss1 libxt6t64 libxtst6 libxv1 libxxf86dga1 libxxf86vm1
lto-disabled-list make
 mesa-libgallium mesa-vulkan-drivers perl-openssl-defaults pkg-config pkgconf pkgconf-
```

```
bin r-base-core r-base-dev
  r-base-html r-cran-boot r-cran-class r-cran-cluster r-cran-codetools r-cran-foreign
r-cran-kernsmooth
  r-cran-lattice r-cran-mass r-cran-matrix r-cran-mgcv r-cran-nlme r-cran-nnet r-cran-
rpart r-cran-spatial
   r-cran-survival r-doc-html r-recommended unzip x11-common x11-utils x11-xserver-
utils xauth xdg-utils zip
  zlib1g-dev zutty
Suggested packages:
   binutils-doc gprofng-gui cpp-doc gcc-13-locales cpp-13-doc debian-keyring g++-
multilib g++-13-multilib
   gcc-13-doc gcc-multilib autoconf automake libtool flex bison gdb gcc-doc gcc-13-
multilib gdb-x86-64-linux-gnu
  gfortran-multilib gfortran-doc gfortran-13-multilib gfortran-13-doc libcoarrays-dev
libdigest-hmac-perl
  libgssapi-perl liblapack-doc bzr libio-compress-brotli-perl icu-doc libcrypt-ssleay-
perl liblzma-doc ncurses-doc
   readline-doc libstdc++-13-doc tcl8.6 tk8.6 libsub-name-perl libbusiness-isbn-perl
libregexp-ipv6-perl
  libauthen-ntlm-perl libunicode-map8-perl libunicode-string-perl xml-twig-tools make-
doc debhelper elpa-ess
   r-doc-info | r-doc-pdf r-mathlib texlive-base texlive-latex-base texlive-plain-
generic texlive-fonts-recommended
   texlive-fonts-extra texlive-extra-utils texlive-latex-recommended texlive-latex-
extra texinfo mozilla
  | www-browser mesa-utils nickle cairo-5c xorg-docs-core
Recommended packages:
  luit
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2 bzip2-doc
cpp cpp-13
 cpp-13-x86-64-linux-gnu cpp-x86-64-linux-gnu dpkg-dev fakeroot fontconfig fontconfig-
config fonts-dejavu-core
  fonts-dejavu-mono g++ g++-13 g++-13-x86-64-linux-gnu g++-x86-64-linux-gnu gcc gcc-13
gcc-13-base
   gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu gfortran gfortran-13 gfortran-13-
x86-64-linux-gnu
  gfortran-x86-64-linux-gnu icu-devtools libalgorithm-diff-perl libalgorithm-diff-xs-
perl libalgorithm-merge-perl
  libasan8 libatomic1 libauthen-sasl-perl libbinutils libblas-dev libblas3 libbz2-dev
libcairo2 libcc1-0
 libclone-perl libctf-nobfd0 libctf0 libdata-dump-perl libdatrie1 libdeflate0 libdpkg-
perl libdrm-amdgpu1
 libdrm-intel1 libegl-mesa0 libegl1 libencode-locale-perl libfakeroot libfile-basedir-
perl
   libfile-desktopentry-perl libfile-fcntllock-perl libfile-listing-perl libfile-
mimeinfo-perl libfont-afm-perl
  libfontconfig1 libgbm1 libgcc-13-dev libgfortran-13-dev libgfortran5 libgl1 libgl1-
mesa-dri libgles2 libglvnd0
 libglx-mesa0 libglx0 libgomp1 libgprofng0 libgraphite2-3 libharfbuzz0b libhtml-form-
perl libhtml-format-perl
   libhtml-parser-perl libhtml-tagset-perl libhtml-tree-perl libhttp-cookies-perl
libhttp-daemon-perl
  libhttp-date-perl libhttp-message-perl libhttp-negotiate-perl libhwasan0 libice6
libicu-dev libio-html-perl
  libio-socket-ssl-perl libio-stringy-perl libipc-system-simple-perl libisl23 libitm1
libjbig0 libjpeg-dev
  libjpeg-turbo8 libjpeg-turbo8-dev libjpeg8 libjpeg8-dev liblapack-dev liblapack3
liblerc4 libllvm20 liblsan0
   liblwp-mediatypes-perl liblwp-protocol-https-perl liblzma-dev libmailtools-perl
```

```
libmpc3 libncurses-dev
      libnet-dbus-perl   libnet-http-perl   libnet-smtp-ssl-perl   libnet-ssleay-perl
libpango-1.0-0 libpangocairo-1.0-0
  libpangoft2-1.0-0 libpaper-utils libpaper1 libpciaccess0 libpcre2-16-0 libpcre2-32-0
libpcre2-dev
   libpcre2-posix3  libpixman-1-0  libpkgconf3  libpng-dev  libpng-tools  libquadmath0
libreadline-dev libsframe1
  libsharpyuv0 libsm6 libstdc++-13-dev libtcl8.6 libthai-data libthai0 libtie-ixhash-
perl libtiff6
  libtimedate-perl libtk8.6 libtry-tiny-perl libtsan2 libubsan1 liburi-perl libvulkan1
libwayland-client0 libwebp7
  libwww-perl libwww-robotrules-perl libx11-6 libx11-data libx11-protocol-perl libx11-
xcb1 libxau6 libxaw7
  libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-render0 libxcb-shape0
libxcb-shm0 libxcb-sync1
   libxcb-xfixes0  libxcb1  libxcomposite1  libxcursor1  libxdmcp6  libxext6  libxfixes3
libxft2 libxi6 libxinerama1
   libxkbfile1  libxml-parser-perl  libxml-twig-perl  libxml-xpathengine-perl  libxmu6
libxmuu1 libxpm4 libxrandr2
  libxrender1 libxshmfence1 libxss1 libxt6t64 libxtst6 libxv1 libxxf86dga1 libxxf86vm1
lto-disabled-list make
 mesa-libgallium mesa-vulkan-drivers perl-openssl-defaults pkg-config pkgconf pkgconf-
bin r-base r-base-core
  r-base-dev r-base-html r-cran-boot r-cran-class r-cran-cluster r-cran-codetools r-
cran-foreign r-cran-kernsmooth
  r-cran-lattice r-cran-mass r-cran-matrix r-cran-mgcv r-cran-nlme r-cran-nnet r-cran-
rpart r-cran-spatial
   r-cran-survival  r-doc-html  r-recommended  unzip  x11-common  x11-utils  x11-xserver-
utils xauth xdg-utils zip
  zlib1g-dev zutty
0 upgraded, 230 newly installed, 0 to remove and 0 not upgraded.
Need to get 224 MB of archives.
After this operation, 770 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Installing software not present in the repositories can be a little more challenging, but usually the software web page can help us. For example, if we want to install RStudio Server in our remote server, we can check the official instructions for Ubuntu at https://docs.posit.co/ide/server-pro/admin/getting_started/installation/install-rstudio-server.html

7.4 Exercise

> **i** Install software
>
> One of you is going to install `btm` from the oficial repositories.

7.5 Removing software

We can remove software from the official repositories with `apt remove`:

```
victor@aula:~$ sudo apt remove btm
[sudo] password for victor:
```

8. Monitoring the system

One important thing we are going to need sometimes is to check the state of the system and the use of resources. We have seen previously a couple of tools to monitorize the disk space use and availability, now we are going to focus on system processes, memory as well as system monitoring (logs)

8.1 `top`, `htop` (and `btm`)

There is hundreds of *system monitors* in Linux. `top` is the common denominator bewteen distributions (is available in most distributions out of the box), but is a little outdated. `htop` is present in Ubuntu and can be installed as well in other distributions and is a modern version of `top`. But there is others, like `btm`, that we installed in the previous chapter.
If you are going to spend a lot of time looking at this, choose the one that gets done what you want in the easiest way.

> **Note**
>
> To start any of them, you just type the name of the command. They will open in the terminal and in most of them you can also use the mouse to arrange columns and select processes.

> **Tip**
>
> Important concepts:
>
> - RAM use: Virtual (reserved) and Resident (actually in use).
> - Swap: Especial reserved disk space intended to be use as RAM.
> - CPU use: usually expressed as percentage.
> - I/O status: Among other things, disk write and read rates.
> - User owning the process.

At the end of the day, what you need is something that tells you how much memory that user is hoarding, how cluttered the I/O is and how much swap is in use to know why the server is not working as before. These tools give you exactly that.

8.2 Exploring the logs

Other part of monitoring the system is being able to tell what happened, not only what is happening. For that, system processes usually maintain logs (text files detailing the activity).

8.2.1 Diagnositcs (`dmesg`)

`dmesg` (**d**iagnostic **mes**sa**g**es) is the main log of the system. It prints the kernel message buffer, so it contains information about all system.

```
victor@aula:~$ sudo dmesg
victor@aula:~$ sudo dmesg --level=err+
```

8.2.2 Logins (`/var/log/auth.log`)

Login attempts (succesful or not) are usually logged at `/var/log/auth.log`. We can use `cat` to print the log on the console:

```
victor@aula:~$ sudo cat /var/log/auth.log
```

> Tip
>
> One of the wonders of Linux is that commands are **pipeable**. The previous log is very long, and is difficult to focus in the important part. We can pipe the result of `cat` to `grep`, a tool that allows to find text in a way that this
>
> ```
> sudo cat /var/log/auth.log | grep victor
> ```
>
> will show only lines with `victor` in it.

8.2.3 SystemD journaling

This is an introductory course, so we are not going to get deep in SystemD. For now we can say that is the first process started when we turn on the server and all other processes are children of this first one. These means that most of the services we have available in the system are launced by systemd, and hence, their logs can be explored with sytemd tools like `systemctl` and `journalctl`. Let's see an example with the ssh.service (the one in charge of launching the SSH server we use to connect).

First, the status of the service

```
victor@aula:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/usr/lib/systemd/system/ssh.service; disabled; preset: enabled)
     Active: active (running) since Tue 2026-06-30 11:02:56 CEST; 1 day 7h ago
TriggeredBy: ● ssh.socket
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 123079 (sshd)
      Tasks: 1 (limit: 38252)
     Memory: 3.7M (peak: 56.7M)
        CPU: 1min 6.437s
     CGroup: /system.slice/ssh.service
             └─123079 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Jul  01  18:14:09  aula  sshd[145233]:  pam_unix(sshd:auth):  authentication  failure;
logname=>
Jul  01  18:14:10  aula  sshd[145233]:  Failed  password  for  invalid  user  ftp1  from
103.52.115>
Jul  01  18:14:11  aula  sshd[145233]:  Received  disconnect  from  103.52.115.76  port
34424:11:>
Jul 01 18:14:11 aula sshd[145233]: Disconnected from invalid user ftp1 103.52.115.76
por>
Jul 01 18:14:30 aula sshd[145240]: Invalid user gitlab-runner from 210.183.21.53 port
43>
Jul 01 18:14:30 aula sshd[145240]: pam_unix(sshd:auth): check pass; user unknown
Jul  01  18:14:30  aula  sshd[145240]:  pam_unix(sshd:auth):  authentication  failure;
logname=>
Jul  01  18:14:32  aula  sshd[145240]:  Failed  password  for  invalid  user  gitlab-runner
from 2>
Jul  01  18:14:33  aula  sshd[145240]:  Received  disconnect  from  210.183.21.53  port
43112:11:>
Jul  01  18:14:33  aula  sshd[145240]:  Disconnected  from  invalid  user  gitlab-runner
210.183.>
lines 1-23/23 (END)
```

And now the logs (journal)

```
victor@aula:~$ sudo journalctl -u ssh
Jul  01  18:14:09  aula  sshd[145233]:  pam_unix(sshd:auth):  authentication  failure;
```

```
logname=>
Jul 01 18:14:10 aula sshd[145233]: Failed password for invalid user ftp1 from
103.52.115>
Jul 01 18:14:11 aula sshd[145233]: Received disconnect from 103.52.115.76 port
34424:11:>
Jul 01 18:14:11 aula sshd[145233]: Disconnected from invalid user ftp1 103.52.115.76
por>
Jul 01 18:14:30 aula sshd[145240]: Invalid user gitlab-runner from 210.183.21.53 port
43>
```

8.3 Other useful tools

- `lsblk` to list block devices (physical disks)
- `lsusb` to list USB devices (this sometimes include network cards and others)
- `lshw` to list hardware in the server
- `ip` to list network devices