

MNXB01 2022 Help Manual

Author: Florido Paganelli florido.paganelli@hep.lu.se

Last updated: 2022-09-03

Table of Contents

1. How to read this document.....	3
2. Brief description of the course environment.....	3
2.1 Basics of authentication.....	5
3. Connecting to Lunarc Aurora Cluster.....	7
3.1 Prerequisites.....	7
3.1.1 Get a SUPR account and a username and password.....	7
3.1.2 Install PocketPass authenticator in your phone.....	8
3.2 Connecting using the ThinLinc client.....	8
3.2.1 In case of emergency, the F8 key.....	9
3.2.2 Sessions.....	9
3.3 Transferring data from your computer to the Aurora cluster.....	10
3.3.1 WinSCP.....	10
3.3.2 FileZilla.....	13
4. Tools on the HPC desktop.....	18
4.1 Overview of the graphical environment.....	18
4.1.1 Mouse actions and Shortcuts.....	18
4.2 The terminal.....	19
4.2.1 Shortcuts.....	20
4.3 The file browser (Caja).....	21
4.3.1 Shortcuts.....	26
4.4 A simple text editor: Pluma.....	27
4.4.1 Editing and saving your first file.....	28
4.4.2 More on opening existing files with Pluma.....	29
4.4.3 Shortcuts.....	30
4.4.4 Customizing Pluma for the course.....	31
4.4.4.1 Create a menu shortcut for Pluma.....	31
4.4.4.2 Configure some pluma options that are useful for coding.....	33
5. Command Line Quick Reference.....	35
5.1 Command line syntax.....	35
5.2 Terminology and special characters.....	36
5.3 Useful commands.....	37
A. Advanced topics.....	39
A.1. Sharing files live between your computer and Aurora using the ThinLinc client.....	39
A.2. The Linux Filesystem.....	44
A.3. An IDE: codeblocks.....	45
A.3.1. Visualizing hidden characters.....	47
A.4. Text-only connection: Using a terminal emulator (Optional).....	50

A.4.1. On Windows.....	50
A.4.2. On Linux or MacOS.....	51
A.5. Command line file transfers.....	51
A.6. SSH key pairs.....	52
A.6.1. Using SSH key pairs instead of username and password.....	54
B. Tips for installing your own Linux.....	56
C. Glossary of terms.....	57
D. References.....	61
E. ChangeLog.....	62

1. How to read this document

This manual contains information on how to configure and use the tools required for the MNXB01-2022 course.

It will help you setting up the course environment for assignments in section 2 to 4.

It should also be used

- as a reference for the command line terminal, see section 5 Command Line Quick Reference
- as a reference for terms you may not have heard before, see section C Glossary of terms.

This manual will be updated as the course progresses with useful chapters to make your journey easy. Updates will be announced on Canvas and changes will be recorded in section E ChangeLog.

2. Brief description of the course environment

For this course we will use the Aurora *cluster* at Lunarc [aurora].

A *computing cluster* is a certain number of computers sitting in a *data center* or *computer hall*, all interconnected within them, that allow for extremely powerful scientific computations, way more powerful than the best computer you ever had.

Most clusters in the world usually run Linux and allow for *command-line, text-only SSH* connections if you do not need a graphical interface. This is **the most common way** to access a cluster. Like many others, also Aurora can be accessed via a *SSH terminal client application*.

However, the world is changing and for some applications is preferable to have graphical environments. For the scope of this course is much more practical to have a graphical environment. Luckily the Aurora cluster is equipped with some graphical *frontend* that allows for a virtual remote desktop, called **HPC (High Performance Computing) desktop**, to interface with the cluster. It is based on a technology called ThinLinc by a company called Cendio.

A *client application* must be installed in your device in order to access the *services* (sometimes referred as *servers*) described above. In this course we will use:

- Mainly use the **ThinLinc** client for accessing the HPC desktop. This can also be used for limited sharing of your device files or folders with the cluster.
- For bulk data transfer we will use a tool called **WinSCP** or other tools available for your operating system (see also section 3.3 Transferring data from your computer to the Aurora cluster).
- (Optional) For command-line connections you can use any SSH client available for your device. A list of such clients is in section A.4 Text-only connection: Using a terminal emulator (Optional).

2. Brief description of the course environment

For enhanced security, Lunararc requires two factor authentication or 2FA. This involves using your mobile phone to prove to the cluster that your *access credentials* (username and password) were actually sent by you and not by some hacker who stole them from you. For 2FA to work, every authentication device (such as your mobile phone) must be *registered* to some authentication provider. Lunararc uses **Phenixid.net** as their main authentication provider, and you will need an app on your smartphone called **PocketPass** to access the services. This application will give you a number called OTP (One Time Password) that you will have to enter at every login.

The procedure you will do everytime a tutorial starts is to login to Aurora, this is:

1. Connect to the Aurora cluster using the ThinLinc Client
2. Authenticate yourself with the Lunararc username and password that either Lunararc or a course teacher will give to you
3. Insert the OTP obtained from the PocketPass app on your mobile.

Illustration 1 shows how all those pieces come together.

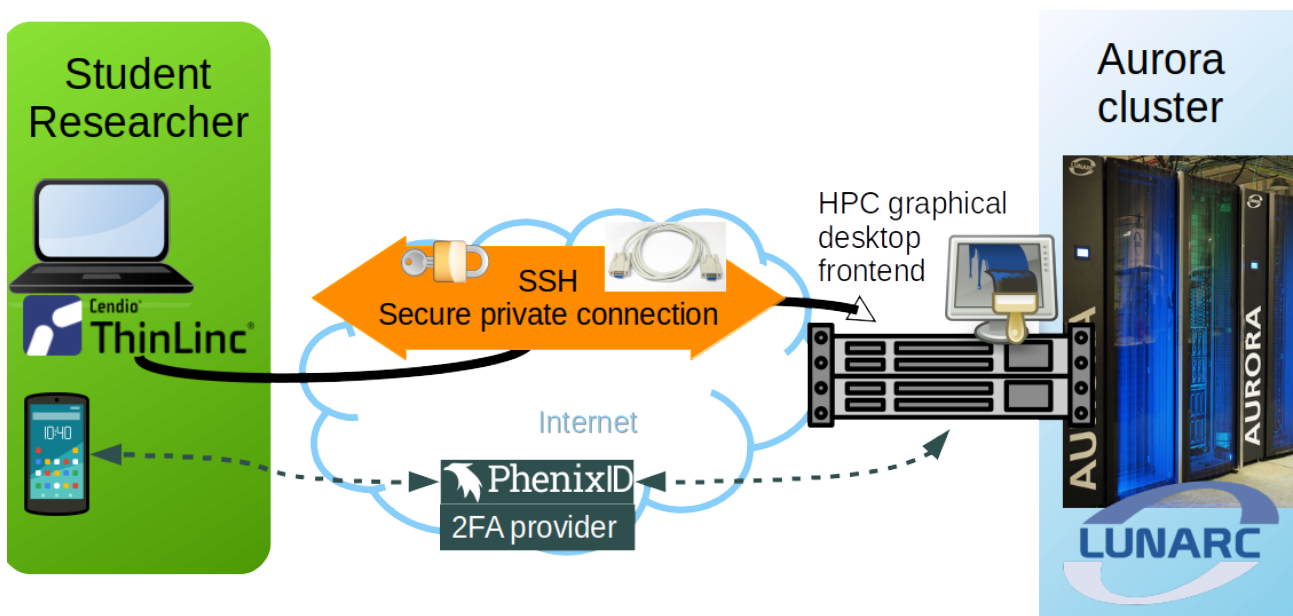


Illustration 1: Tools involved in the access to Lunararc cluster

2.1 Basics of authentication

Remember: your password is **private** and **must not be disclosed to others** at any rate. In case the password gets stolen, you put at risk the entire infrastructure it is used to access.

You will not find some of the information that follows in Lunarc documentation for practical reasons. But in this course we try to give the needed tools for you to find your way through the complicated ever changing world of IT. Lots of things change in IT, but the basic concepts remain. You just don't see them explained often.

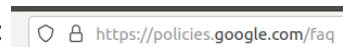
Authentication is the process with which two or more parties authenticate to each other, that is, present to each other some kind of *proof of identity*. Authentication is based on *trust*.

Google used to trust that you are the actual “you” when you log in with your *username* and *password*. Since these two can be stolen, nowadays Google ask for your mobile phone or other details.

Have you ever asked yourself how do you know Google is Google? I give you the answer: you don't. You *trust* (or rather, you *hope*) Google will be at www.google.com. The trust is based on the simple fact that website names are universally unique and cannot be shared, and Google has enough money to own www.google.com virtually forever. Shall we call it trust by power?

In general, very little is done for you to trust the server. Much more is done for the server to trust you.

Today's trust in Google is taken care by your browser, that carries some special certificates to identify that www.google.com is actually a legit website created by Google LLC. Google identifies itself by using a certificate. For example in the Firefox browser you can get info about this certificate by clicking on the small lock icon next to the website URL:



The lock also means that the connection between you and the website is securely encrypted, that is, the data flow between your browser and google can only be read and understood by google and your browser, and nothing else. Everybody else trying to steal such data will be only able to see a seemingly confusing flow of data. This is also implied by the s in https in the website address – http *secure*. http is the communication *protocol* with which webpages are sent all over the world. https encrypts the communication channel with a technology called SSL that is beyond the scope of this course.

Going back to the HPC world, the secure protocol in use is SSH. SSH stands for *Secure* SHell.

- Secure: authenticated and encrypted
 - authenticated: the cluster knows who your are and you can be sure the cluster is the one you want to connect to. You both know your **identities**

2. Brief description of the course environment

- encrypted: encoded in such a way that the information can only be read by you and the cluster, achieving what is called **confidentiality**
- SHell: a kind of command line interface, a way for humans to interact with computers by writing *commands*

As you might have understood from the Google example, the **first** thing one needs to answer when connecting to a service is: is this service *actually* the one I am connecting to?

In SSH you can identify a server via a strange sequence of characters that uniquely identifies that machine. It is called a **hash**. A typical SSH server hash looks like this:

```
ECDSA MD5:e9:bf:d5:af:f1:99:a6:02:90:03:b0:f3:1a:87:61:a2
ECDSA SHA256:99/VwIhxK0S26BxBCXuPt7KUWOs2V1GxsmzWnY02Up0
```

MD5 and SHA256 are *hashing algorithms* used to generate the hash. Nowadays SHA256 is more secure, but for compatibility reason some systems still show the MD5 format.

ECDSA is the algorithm used to create the server digital identity. Other common algorithms are: RSA, ED25519. We will not cover this or any cryptography in the course, but it is mentioned here so that you know what that is about when you'll see it.

Normally, a system administrator should tell you which hash to trust. For some reason this is rarely done. In this course we will do it properly and I will tell you what is the identity of the Aurora cluster you can trust.

The **second** thing one needs to do is to tell the cluster one's identity by sending it username and password. It's the server asking you: who are you?

Today's security standards require a **third** thing one needs to do. Somebody could have stolen your username and password. The server (the cluster) wants to be sure you are who "you" claim to be.

This is achieved with MFA (Multi Factor Authentication) a technique based on the fact that you need to authenticate yourself with something *about yourself at the moment of authentication*.

In reality this is just your mobile phone or some electronic device or app that generates some value that lasts only a few seconds, the very moment when you try to login. Such short lived value is usually called **OTP, One Time Password**. The provider of the OTP password is *trusted* by the server that you're trying to connect to.

So to recap, authentication involves these three steps:

- 1) **Identify a remote server** or service (Is the server *actually* the server?)
- 2) **Identify yourself** to the service (*who* am I on that server?)
- 3) **Prove your identity** to the service (am I *really* who I claim to be?)

My personal suggestion for your career is that whenever you're asked to login via SSH to a system, you ask the system administrator in charge to tell you **what is the SSH hash of the host**.

3. Connecting to Lunarc Aurora Cluster

3.1 Prerequisites

3.1.1 Get a SUPR account and an Aurora username and password

A SUPR account will give you the possibility to request computing resources within all the the National Swedish Infrastructure for Computing (SNIC) computing centers, a federation of which Lunarc is member.

1) **Create a SUPR account** at

<https://supr.snic.se/>

First, try to click on “Login using SWAMID”. You should be able to login as a LU student.

If you cannot login, click on “Register New Person” and then try to “Register via SWAMID”.

If that doesn’t work you can “Register without SWAMID” but it may take time, as the process involves sending a paper form via post service.

At the end of the process you may have a SUPR password. Such password is NOT the one used to access the cluster. Do not confuse the two. This password is to access the SUPR services which are provided to researchers across all Sweden.

Upon registration it is very important that you provide this information:

- Your correct **First Name** and **Last Name**.
- A **valid email address**, the student one from LU or other Swedish university is preferred. Providing a different email address may cause delays in your approval, if not rejection. The same address will be used by Lunarc to provide the first login information.
- A valid **mobile phone number**. This is required to access the computing services.

2) If the registration succeeds, you should be able to **pick a project**. For this course you should ask for membership to the following project:

LU 2021/7-65

<https://supr.snic.se/project/21026/>

If you’re successfully registered and logged in you can click on the above link and request for membership approval.

Then wait for approval. You will receive an email notification when your membership is approved.

3) Obtain your login credentials (**username** and **password**) from Lunarc. As soon as your registration is successful you will get a notification from Lunarc with the details. A mobile phone is required to login to Aurora.

You can read more about SUPR and SNIC (Swedish National Infrastructure for Computing) at <https://www.snic.se/>

3.1.2 Install PocketPass authenticator in your phone

Install and configure the PocketPass authenticator on your mobile phone (the one you registered in SUPR!). Follow:

https://lunarc-documentation.readthedocs.io/en/latest/authenticator_howto/

3.2 Connecting using the ThinLinc client

Most of the documentation is provided by Lunarc itself. These pages will just add practical information that might be missing.

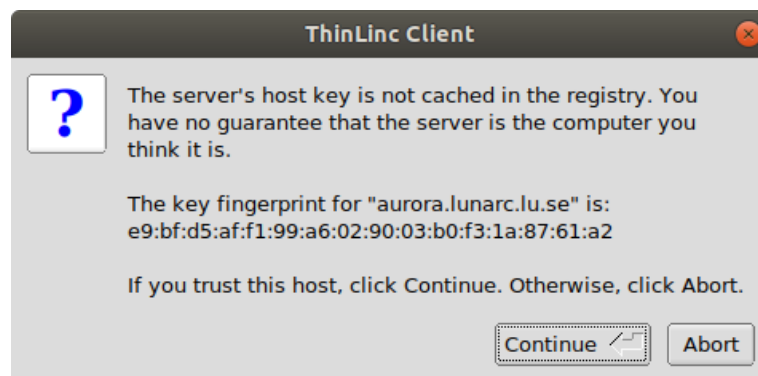
In order to connect you need to complete the prerequisites in section

To install and configure the ThinLinc client on your computer, follow:

https://lunarc-documentation.readthedocs.io/en/latest/using_hpc_desktop/

Suggestion: **Read the whole page at the link above at least once before you start doing any practical thing.**

When logging in the first time, you will be asked whether you trust the server identity:



Verify that the SSH identity hash of the server matches one of the following and click **Continue** if so.

```
ECDSA MD5:e9:bf:d5:af:f1:99:a6:02:90:03:b0:f3:1a:87:61:a2
RSA MD5:6a:e7:89:f5:de:cf:f7:3b:da:3f:cb:bb:9d:62:57:f4
ED25519 MD5:bb:90:61:be:0c:92:ad:c1:3e:b2:a3:08:57:11:db:0c
ECDSA SHA256:99/VwIhxK0S26BxBCXuPt7KUW0s2V1GxsmzWnY02Up0
RSA SHA256:T0HzXcFkoAlyIqWeAwYvPbayhLHApz5i5dBgkEqQPYk
ED25519 SHA256:oUm1L3HujmWyVOpdtpkArAaOgN0PZtkuVjIbIMduyk
```


3.Connecting to Lunararc Aurora Cluster

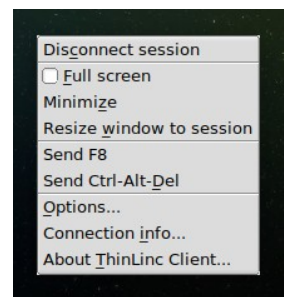
If you successfully manage to login, you should be able to see the HPC desktop:



3.2.1 In case of emergency, the F8 key

The F8 key is used to interact with the ThinLinc client application. It will allow you to go back to your computer own Operating System and customize the way the ThinLinc application behaves.

- I suggest to disable the fullscreen mode at the beginning. Press **[F8]** and untick “Fullscreen”
- To go back to your own computer desktop, press **[F8]** and then choose “Minimize”
- If the screen gets stuck, press **[F8]** and then **[Alt][Tab]**(ALT+TAB)



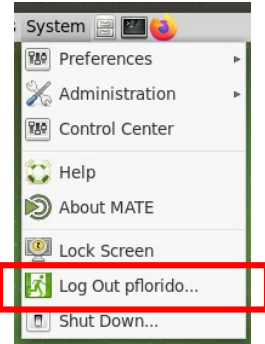
3.2.2 Sessions

Whenever you login to any system that is the start of a **session**. Some session only last the time of your login, some other can survive after you logout. The Lunararc HPC desktop survives your logins if you do not actively logout. It can survive up to **7 days**, then it will be forcibly **closed**. You will lose all the ongoing work and open windows.

- If you just close the ThinLinc application by clicking on “Disconnect session” or by just closing the application window, your session will be kept alive on the frontend.

3.Connecting to Lunar Arc Aurora Cluster

- If you want to **completely close a session**, terminating all the programs you're running on the HPC desktop, you need to actively logout using the remote desktop functions. On the top left bar, click on "System" → "Log Out <your username>..."



3.3 Transferring data from your computer to the Aurora cluster

To transfer data from your computer to Aurora in bulk mode, that is, without logging in using the ThinLinc client, but simply dragging and dropping from your own computer's desktop, the best way is to use a standalone data transfer client such as WinSCP or FileZilla.

The classic way for Linux and MacOSX would be to use the command line tools `scp` or `sftp`. I will briefly document that in the Advanced Topics section.

In what follows we will identify the files present on your own computer as **local files**, while the files that are on Aurora will be called **remote files**. You will see these names also in most programs that do data transfer.

3.3.1 WinSCP

WinSCP is only available for Microsoft Windows. It is recommended by Lunar Arc. You can download WinSCP at this link (and only this one! avoid other sources.)

<https://winscp.net/eng/index.php>

You can find the information about how to configure it here:

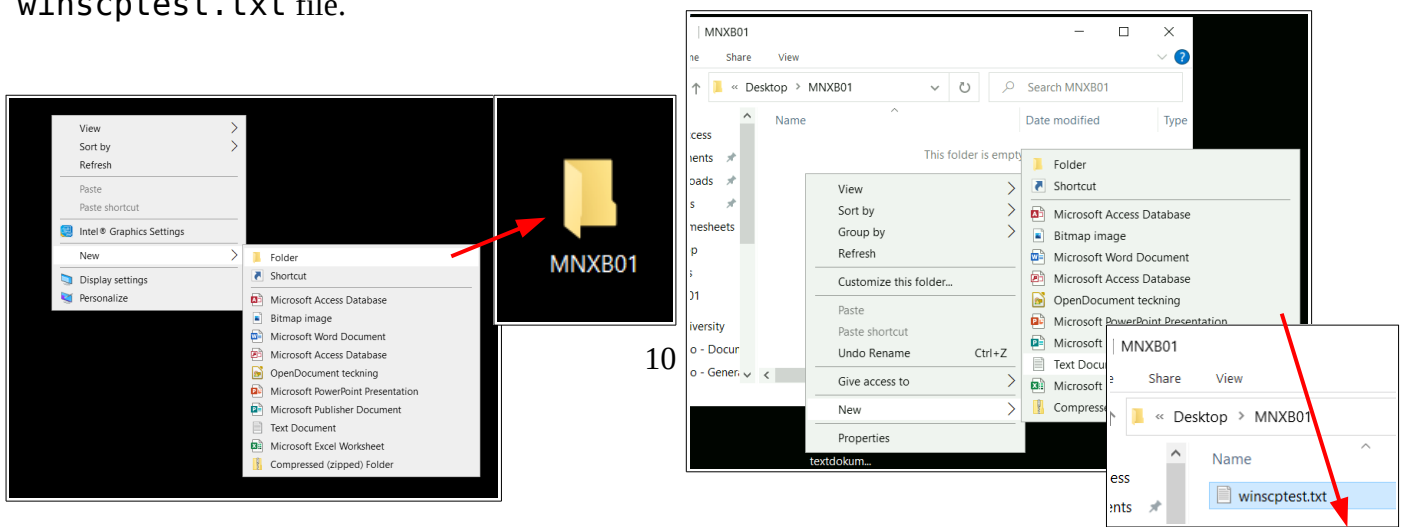
https://lunarc-documentation.readthedocs.io/en/latest/login_howto/#file-transfers

And on WinSCP official documentation:

<https://winscp.net/eng/docs/start>

In what follows I show you an example of how to transfer a file, even though it's quite intuitive.

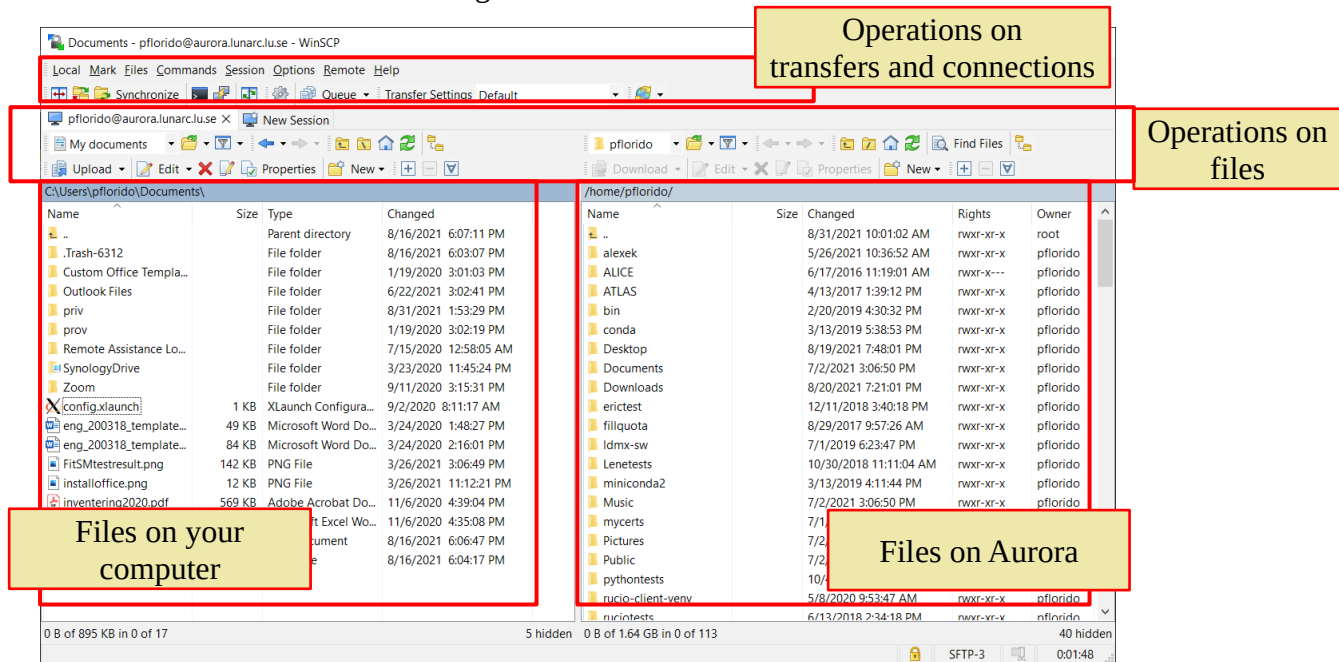
1) Create a folder MNXB01 on the Desktop of your computer. Enter the folder and create a `winscptest.txt` file.



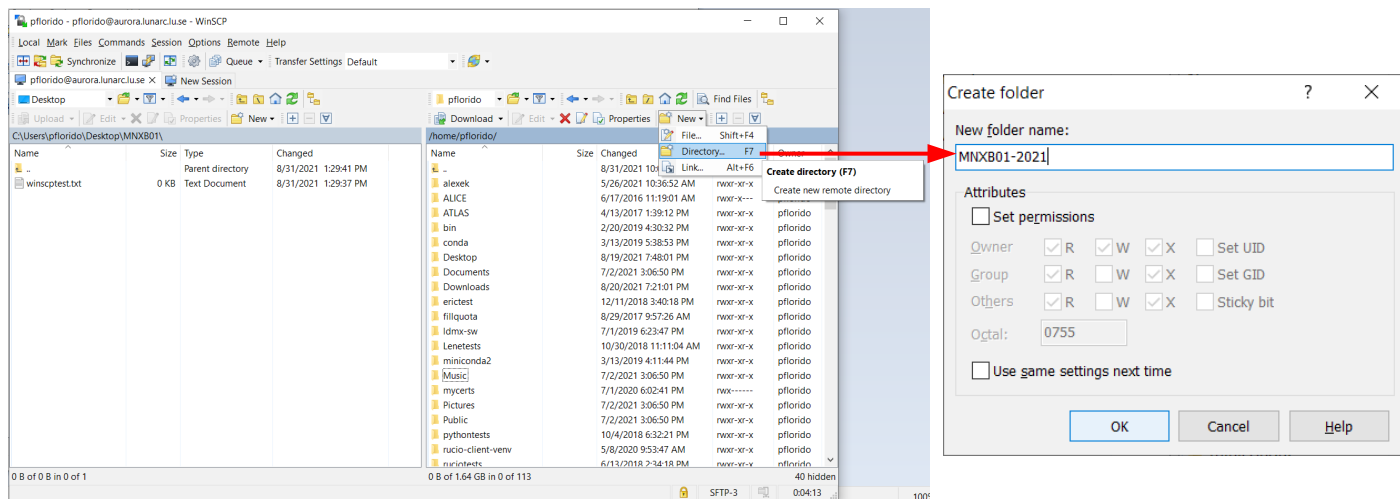
3. Connecting to Lunarc Aurora Cluster

2) Open WinSCP and configure it according to Lunarc documentation. Connect to Aurora by clicking on the Login button.

3) Here's a brief explanation of WinSCP panel. I prefer the commander view, with the local folders on the left and the remote folders on the right.



4) Create a new folder called MNXB01 - 2022 in your Lunarc home (right panel) as in the pictures below (the picture refers to 2021, but please change to the current year) and press OK:



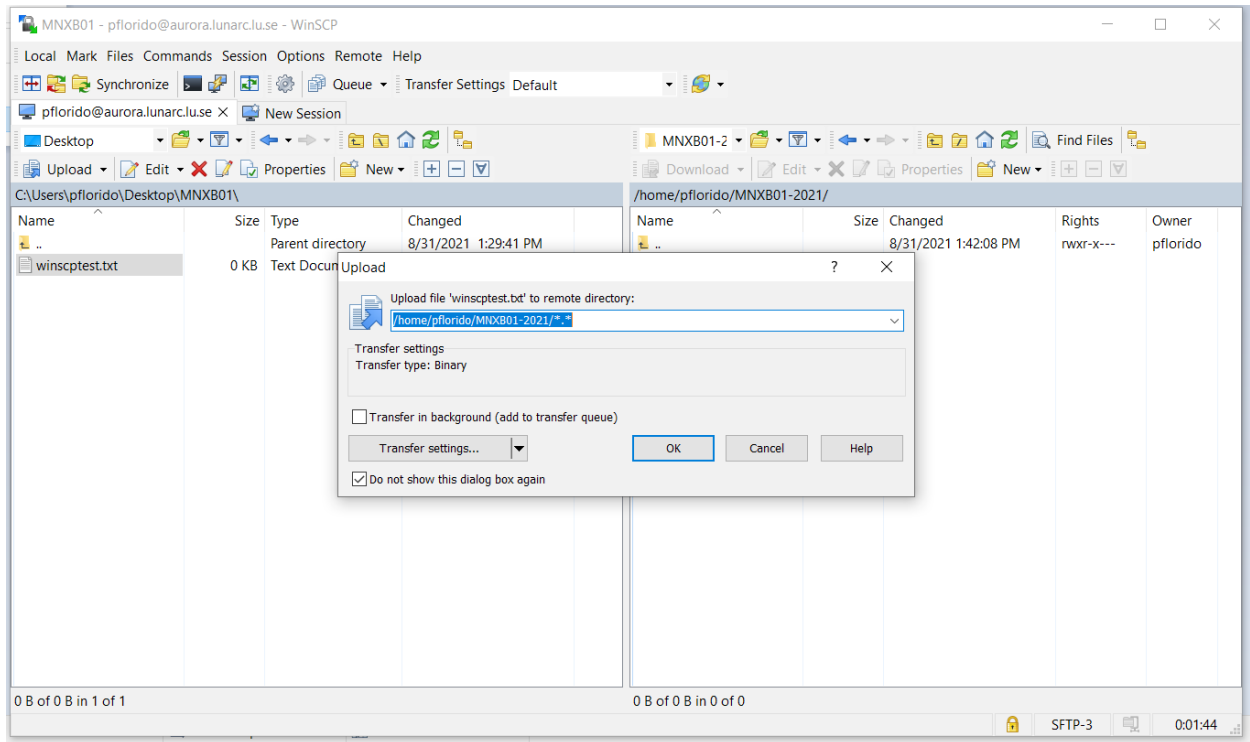
5) To transfer the created file winscptest.txt:

5.1) navigate to your computers' Desktop\MNXB01 folder on the left panel,

5.2) navigate to the newly created MNXB01-2022 folder on Aurora (right panel)

3.Connecting to Lunarc Aurora Cluster

5.3) Drag and drop the winscpctest.txt file from the left to the right panel. Accept the upload panel by pressing OK.



6) The final result should look like this:

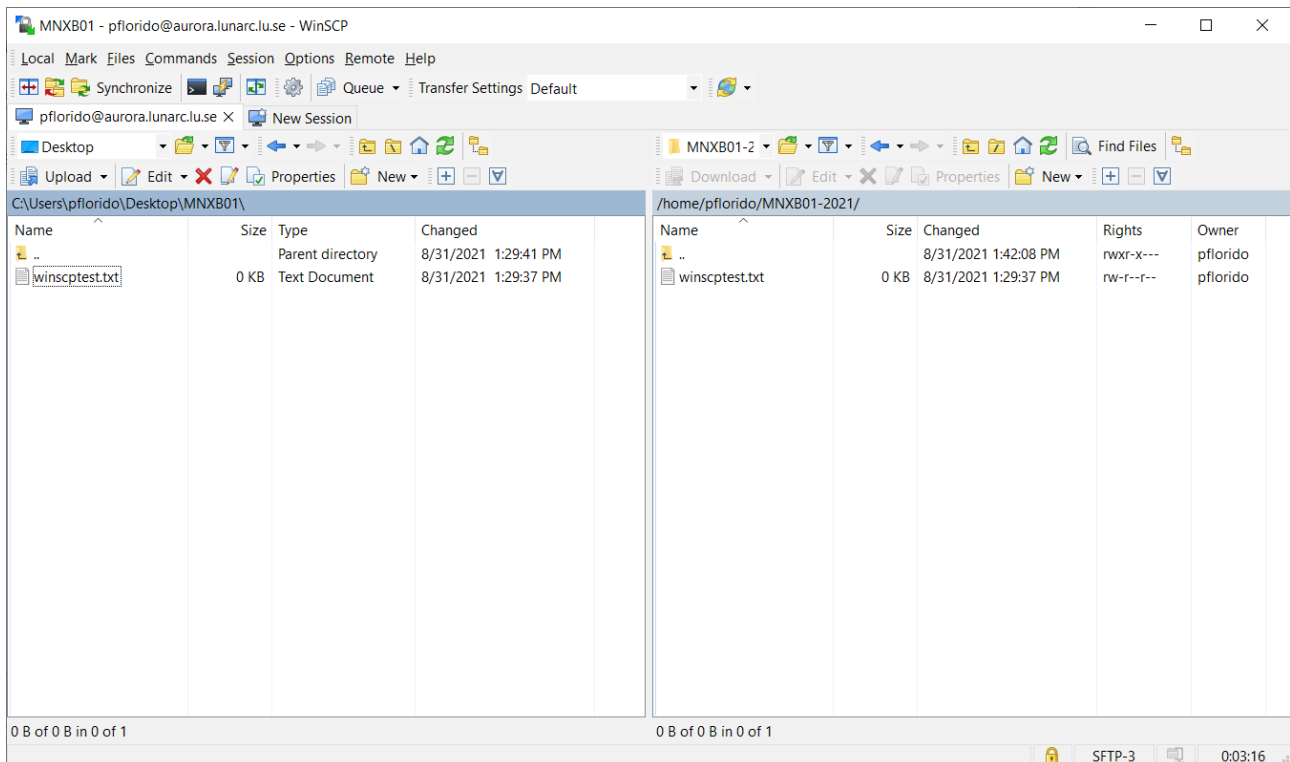


Illustration 2: Final result of a sample WinSCP copy

3.3.2 FileZilla

FileZilla is an open source FTP client that can also do SFTP transfers based on SSH. It is considered less secure than WinSCP, but it's currently maintained and it runs on all operating systems, so I present it here as an alternative for MacOSX and Linux. Moreover their default installer sometimes contains unwanted software, so please follow the instructions I provide below to avoid that.

FileZilla is available for most Linux distributions directly from the distribution repository. Follow your favorite distribution instructions to download it if you're using Linux.

For Windows and Apple you can download FileZilla from their official page at this link. Use only this link, other links on other pages or even their homepage sometimes contain unwanted software for sponsoring purposes.


https://filezilla-project.org/download.php?show_all=1

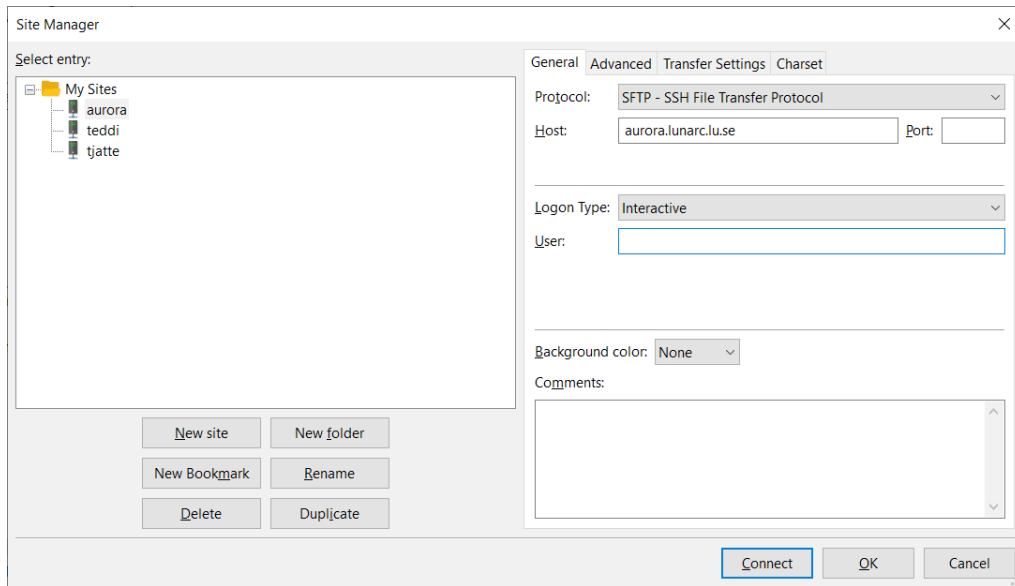
In most cases you need the 64 bit version.

On modern Apple computers with ARM M1 processors, the system will ask you to install the "Rosetta" compatibility layer to be able to run Intel x86 code. Say yes to such request. At the moment there is no Filezilla compiled for M1.

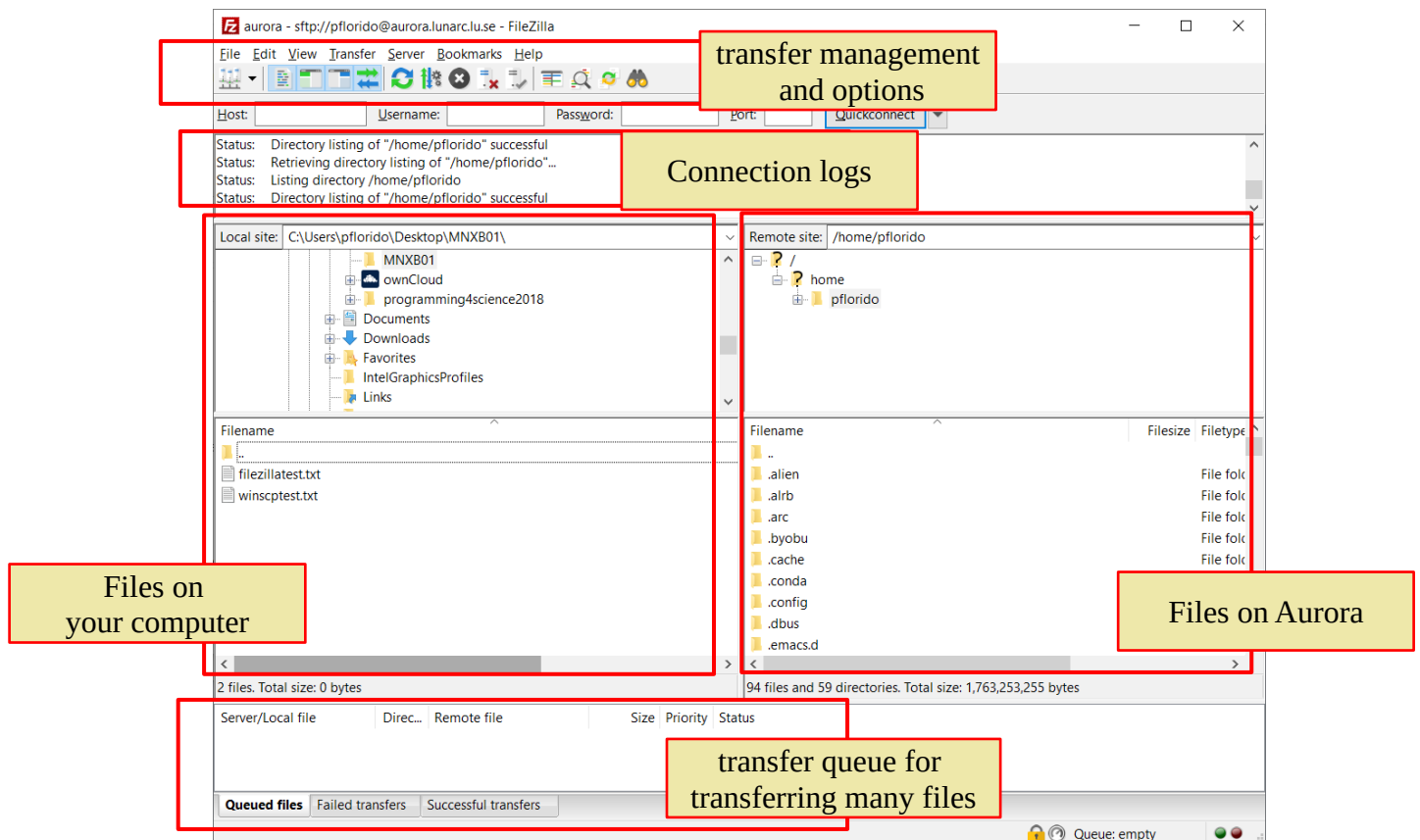
1) Create a folder MNXB01 on the Desktop of your computer. Enter the folder and create a `filezillatest.txt` file. See also step 1 section 3.3.1

3.Connecting to Lunarc Aurora Cluster

2) Configure FileZilla for Aurora. Click on the servers icon  and add a connection to Aurora as shown in the picture. Selecte *SFTP* protocol and make sure to choose *Logon Type: Interactive* as it is required for the 2FA OTP step. When done click on Connect, then enter your username, password and OTP when requested, as in the normal Aurora login steps.

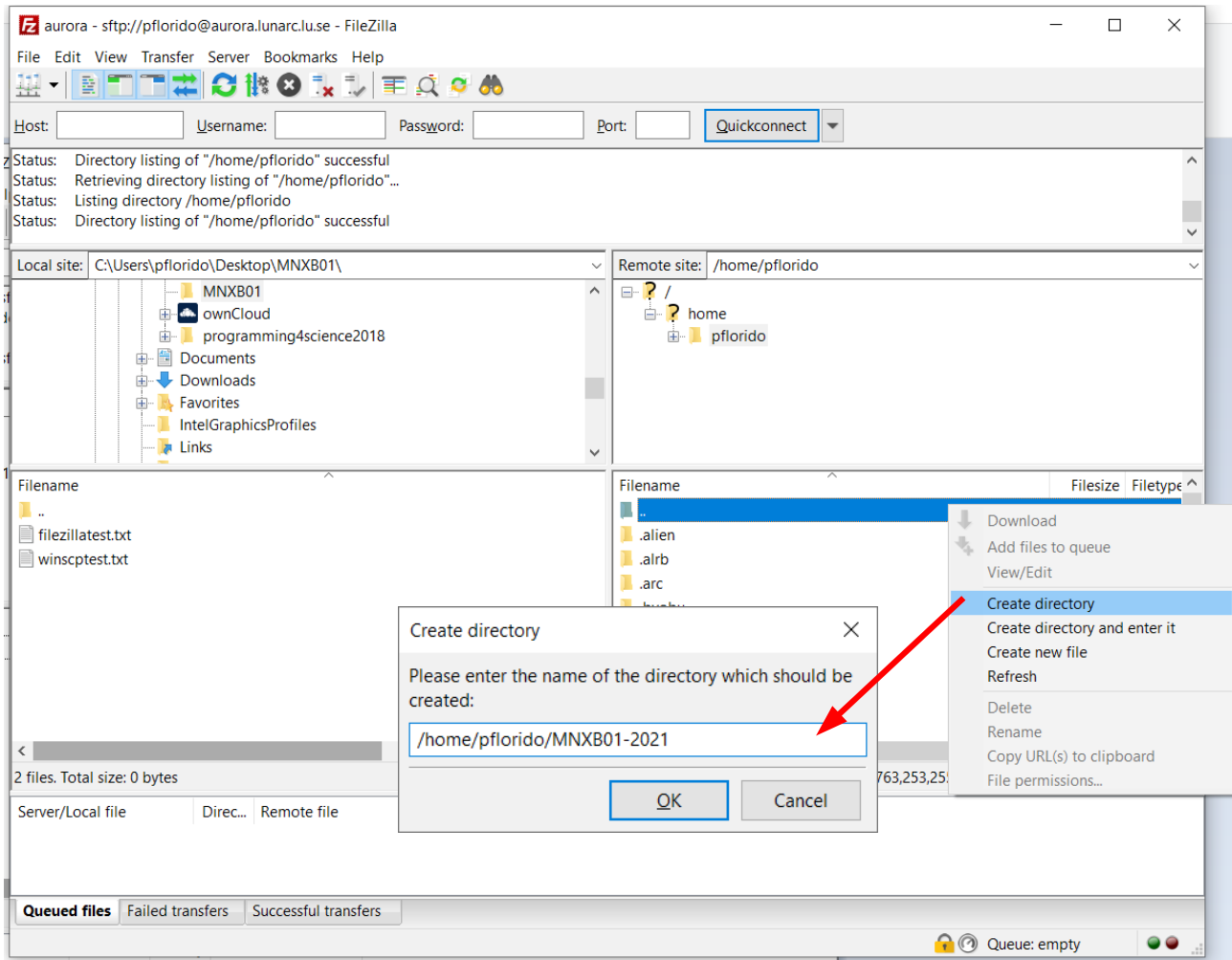


3) Brief explanation of the FileZilla interface:



3.Connecting to Lunarc Aurora Cluster

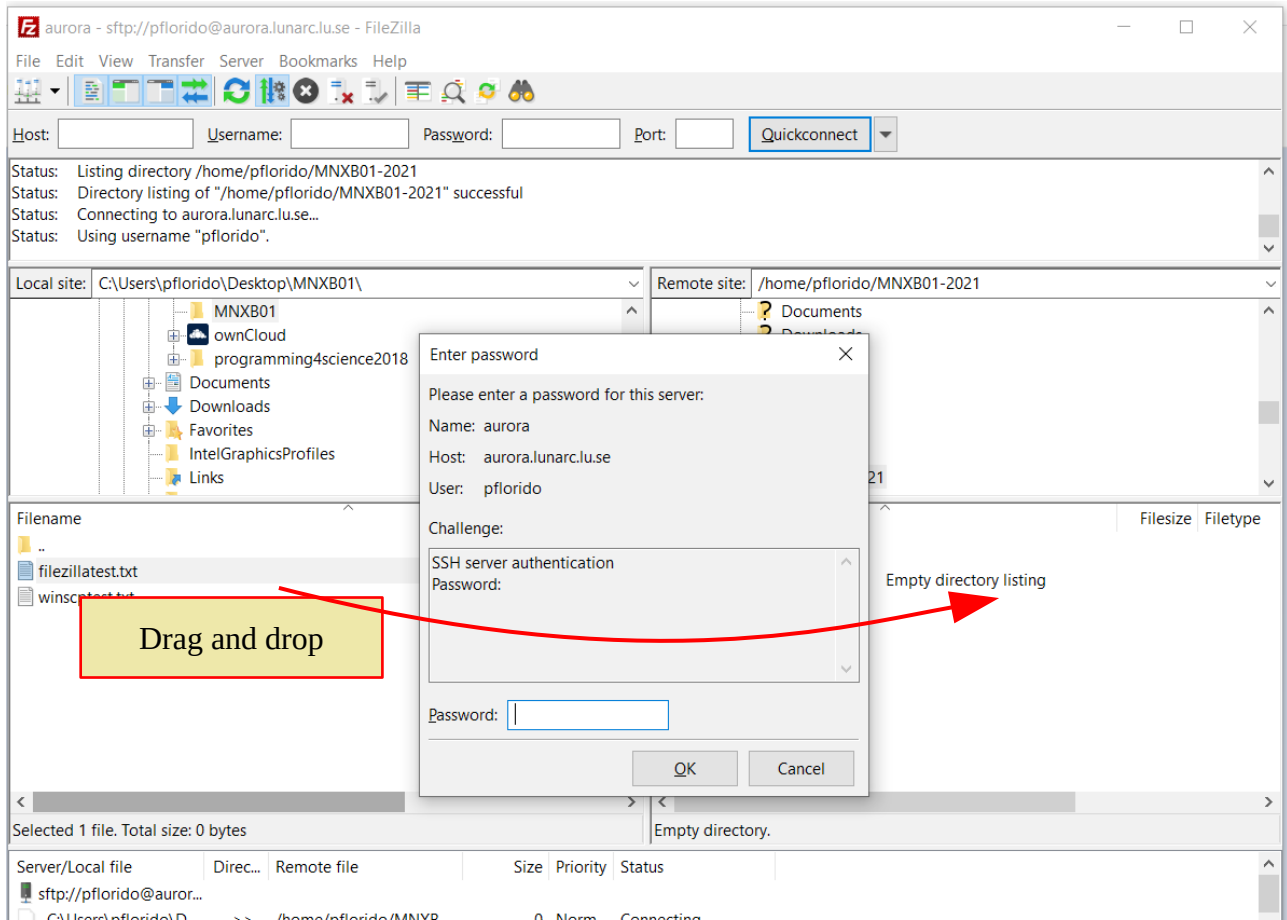
4) Create a new folder called **MNXB01 -2022** on Aurora by right-clicking anywhere on the right panel, as shown in the picture below (the pictures may show 2021 but please change to the current year); then insert the name of the folder when the dialog appears. Then click Ok. In some cases the system may ask to login again, as it disconnects to save bandwidth. Double click on the folder you just created to enter it.



3.Connecting to Lunarc Aurora Cluster

5) Navigate to the folder **MNXB01** on your local computer files (the left panel), the one with the **filezillatest.txt** file. To transfer the file to Aurora, drag and drop it from the left panel to the right.

You will be asked again for login and OTP.



3.Connecting to Lunarc Aurora Cluster

6) If the transfer has completed successfully, it should look like in the picture below.

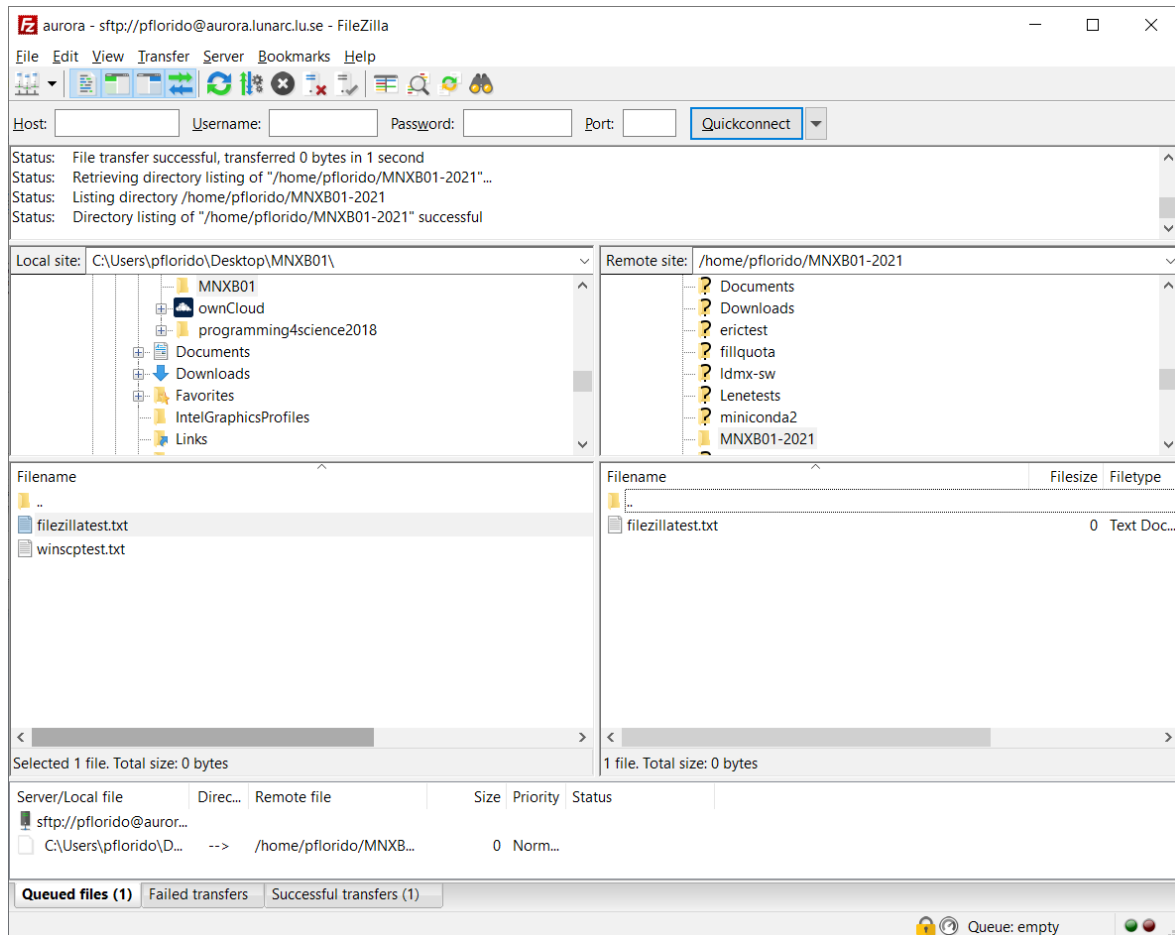


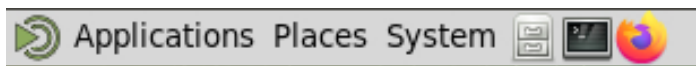
Illustration 3: Final result of a file transfer via FileZilla

4. Tools on the HPC desktop

There are a couple of applications that will be used intensively during the course and that you must learn how to start/stop/interact with.

4.1 Overview of the graphical environment

The HPC desktop graphical environment is one of the many graphical environment available for Linux. This one is based on the window manager called MATE [MATE]. It is supposedly intuitive to use, however to make your life easier we will describe some of its features and handy keyboard shortcuts. It works pretty much like Windows or MacOS.



On the top left you can see the **Menu bar**. *Applications* contains programs, *Places* allows you to browse the files on

the cluster, *System* allows you to logout.

The other icons will be described later, the round one is the Firefox internet browser.



On the top right you can see the **Icon tray**. It contains date and time and information about the language. You can change the language if you wish but I do not recommend it.



On the lower left you can see the **Desktop button**. It brings you back to the desktop view when you click once, and back to your open apps when you click a second time.



On the lower right you can see the **workspace selector**. You have 4 desktops where you can spread you apps. By clicking on the rectangles you switch workspace. You can move apps around

workspaces by right-clicking on the top of the app window.

4.1.1 Mouse actions and Shortcuts

When you are working on some program that program is said to be *in focus* in the graphical environment. The keyboard strokes will be received by the program *in focus*.

Visual objects you can interact with are icons, buttons, windows... pretty much everything you mouse pointer can reach.




: One left click on a visual object selects that visual object.



2x : Double left click on a visual object performs the default action for that object.



: Right click on an any visual object shows a list of possible actions for that object.

MacOS pointer systems have only one button therefore they have special ways to *simulate* the right click. It may involve using two fingers or pressing the special key  together with the left click. It depends on the operating system version and I cannot give an exhaustive list here.

4.Tools on the HPC desktop

The following keyboard shortcuts apply to all applications windows:

  (ALT+TAB) : Cycle focus of open windows/apps, switches from one app to the other

  : Close the application currently in focus

4.2 The terminal


The terminal is the most useful tool in the Unix/Linux world and it is becoming more and more common also for advanced Windows and Apple users.

It is an application that emulates old school text only interfaces, in which the user interacts with the computer by typing *commands*. Commands are special keywords or small applications with a simple purpose. Combining them together allows the user to make the computer do more and more complex tasks.



On the HPC desktop you can open the terminal by clicking the terminal icon on the top bar on the left

The picture below shows some commands. A list of common commands is in section 5 and you will exercise with some of them in one of the tutorials.

To *execute* commands like `ls` or `whoami` in the examples below, type the command and then press  (the *Enter* key)

```

Mate Terminal
File Edit View Search Terminal Help
-rw-rw-r-- 1 pflorido hep 0 Mar 22 2017 testa2
drwxr-xr-x 3 pflorido hep 4096 Feb 4 2021 testalex
-rwxr-xr-x 1 pflorido hep 63 Feb 4 2021 testbatches.
sh
-rwxr-xr-x 1 pflorido hep 115 Sep 21 2016 testbatch.sh
-rwxr-xr-x 1 pflorido sto_ldmx_simulations 218 Feb 19 16:14 testcopy.sh
drwxr-xr-x 4 pflorido hep 32768 Feb 3 2021 testfarm
drwxr-xr-x 5 pflorido hep 4096 Apr 30 2019 testlenc
drwxr-xr-x 3 pflorido hep 4096 Jan 8 2021 tests
-rw-r--r-- 1 pflorido hep 85 Feb 5 2019 #tests.txt#
-rw-r--r-- 1 pflorido hep 95 Feb 19 2019 tests.txt
-rw-r--r-- 1 pflorido hep 94 Feb 19 2019 tests.txt~
drwxr-xr-x 3 pflorido hep 4096 Sep 21 2018 tf2017
lrwxrwxrwx 1 pflorido hep 45 Aug 16 17:59 thindrives -
> /var/opt/thinlinc/sessions/pflorido/21/drives
drwxr-xr-x 3 pflorido hep 4096 Jun 11 2020 venvs
drwxr-xr-x 2 pflorido hep 4096 Jul 2 15:06 Videos
drwxr-xr-x 2 pflorido hep 4096 Dec 4 2017 Wolfram Math
ematica
[pflorido@aurora-rviz01 ~]$ ls Desktop/
whatislove.txt
[pflorido@aurora-rviz01 ~]$ whoami
pflorido
[pflorido@aurora-rviz01 ~]$

```

You can close a terminal by typing the command `exit` or by closing the window.

4.2.1 Shortcuts

You can select text by using your mouse and the left button.

- Ctrl** **+** (Ctrl+plus): Increase font (characters) size
- Ctrl** **-** (Ctrl+minus): Decrease font (characters) size
- Ctrl** **0** (Ctrl+zero): Reset font (characters) size to default
- Ctrl** **⇧** **C** (Ctrl+Shift+c) or : Copy selected text
- Ctrl** **⇧** **V** (Ctrl+Shift+v) or : Paste selected text
- Ctrl** **⇧** **N** (Ctrl+Shift+n): Open a new terminal in a new window
- Ctrl** **⇧** **T** (Ctrl+Shift+t): Open a new terminal in a new tab
- Ctrl** **PageUp** or **Ctrl** **PageDown**: Cycle focus of open tabs

The following shortcuts behavior depends on the shell program. These below work mostly for bash.

TAB key. This beautiful key in Linux helps you autocompleting a command. Start typing a command and hit TAB to autocomplete. When hit without writing any command, it searches

4.Tools on the HPC desktop

through the files in the current directory. In some special cases it is even smart enough to select only the files that a certain command works on.

Home Go to the beginning of the current line

End Go to the end of the current line

← **→** Move left or right on the current line

↑ **↓** Up/Down arrows: scroll the history of commands you typed. Up goes back in history.

Ins Switch insert/edit mode, it works the same as in most editors including microsoft word.

←, **Del** Work pretty much like in any other editor:

← deletes characters on the **left** of the cursor

Del removes characters at the **right** of the cursor.

Ctrl is a special key in the terminal. It is usually represented in documentation with a caret symbol **^**. It is used to send special characters to the terminal that have special meanings.

Here I will just mention the most notable ones **that you have to memorize.**

Ctrl **c** : Usually represented as **^C** . Sends a SIGINT system signal to the current application in foreground. This signal tells the system kernel to terminate gracefully the current application.

NOTE: this is NOT used to copy text in the terminal!

Ctrl **z** : Usually represented as **^Z** . Sends a SIGSTP system signal to the current application in foreground. This signal tells the system kernel to SUSPEND the execution of the current application and return the I/O control to the terminal. **NOTE: this is NOT used to undo changes in the terminal!**

Ctrl **d** : Usually represented as **^D** . Sends the EOF (End Of File) character. This character is usually contained at the end of each file. When pressed on the command line, it is interpreted by the terminal as "stop sending further data" and therefore as a wish to logout and terminate the current SSH connection. **Be careful when you use it: it may log you out of the shell and terminal window!**

4.3 The file browser (Caja)

The file browser is used to graphically browse the files present on the cluster.

It is recommended that you read the section A.2 The Linux Filesystem if you're completely new to filesystems and their concepts. These concepts will be also presented in one of the tutorials.

You can open the filebrowser by clicking on *Places* or the filebrowser icon  in the menu bar.

Clicking on any of the following icons on the desktop brings you to special places in the *directory tree*:



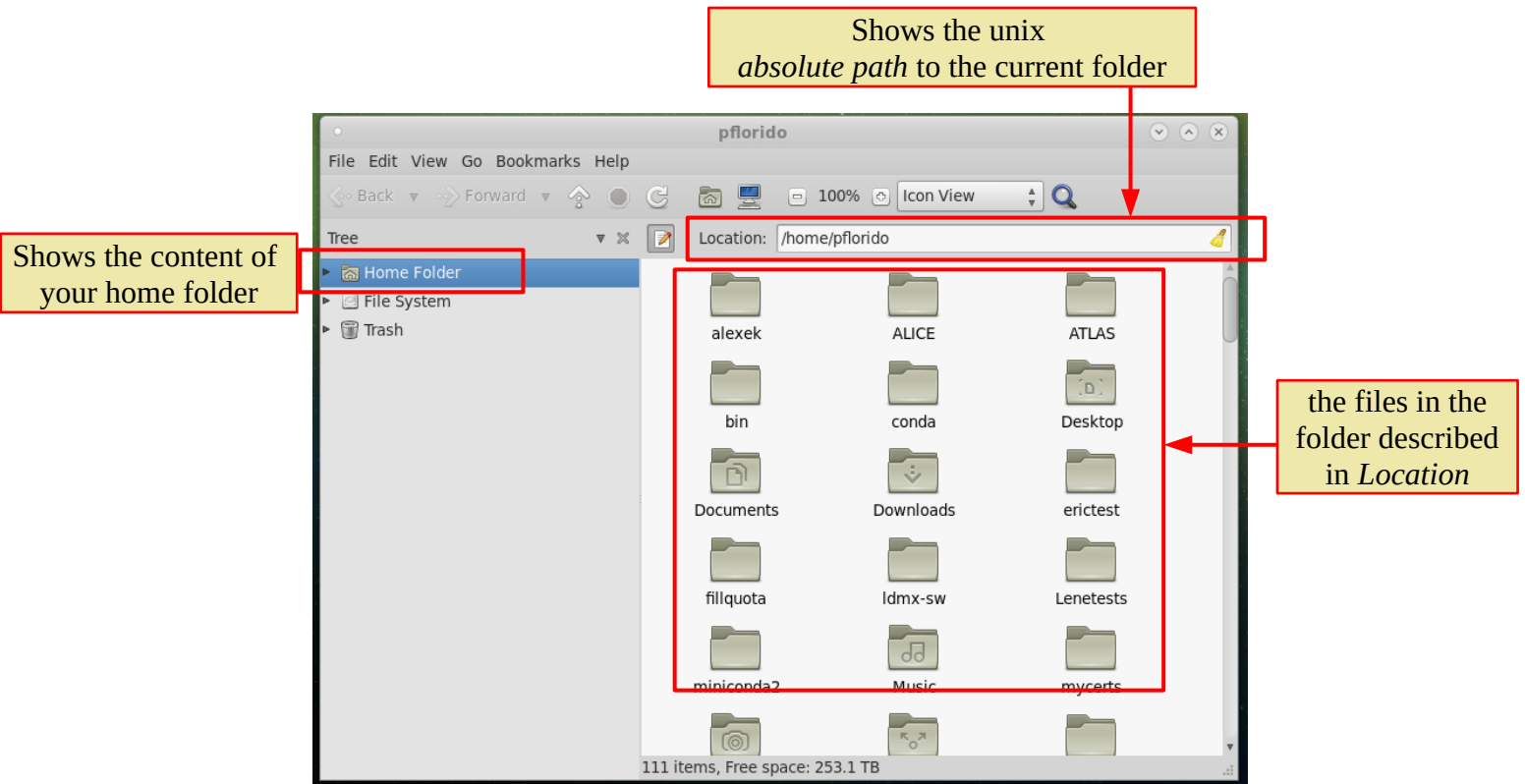
View all connected devices. Doesn't make much sense on a cluster.

Your home folder. This is the place where all your files are.

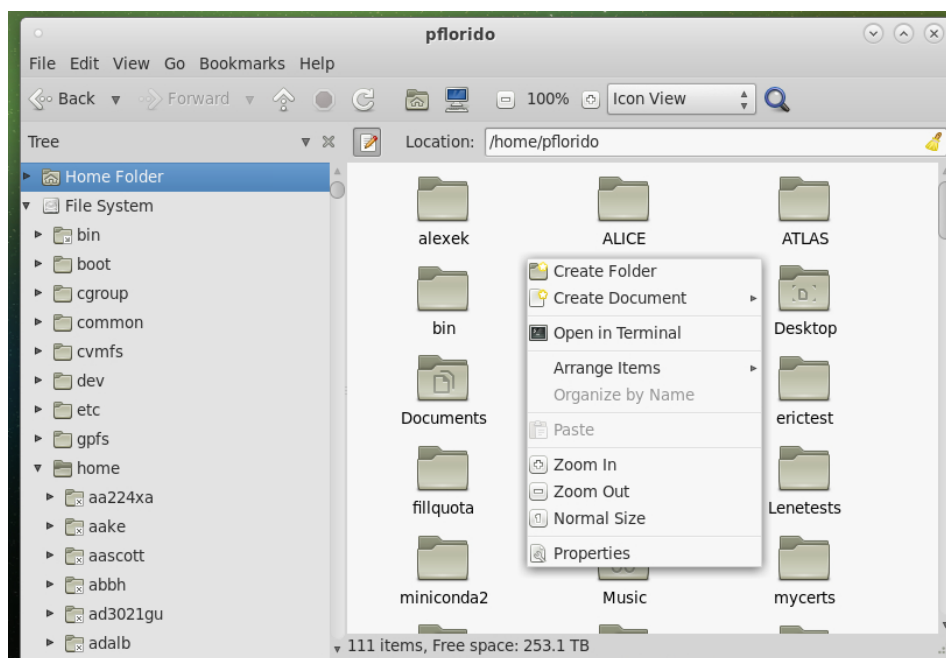
Trashcan. This is where the deleted files go.

4. Tools on the HPC desktop

Here's some highlights about the file browser window:

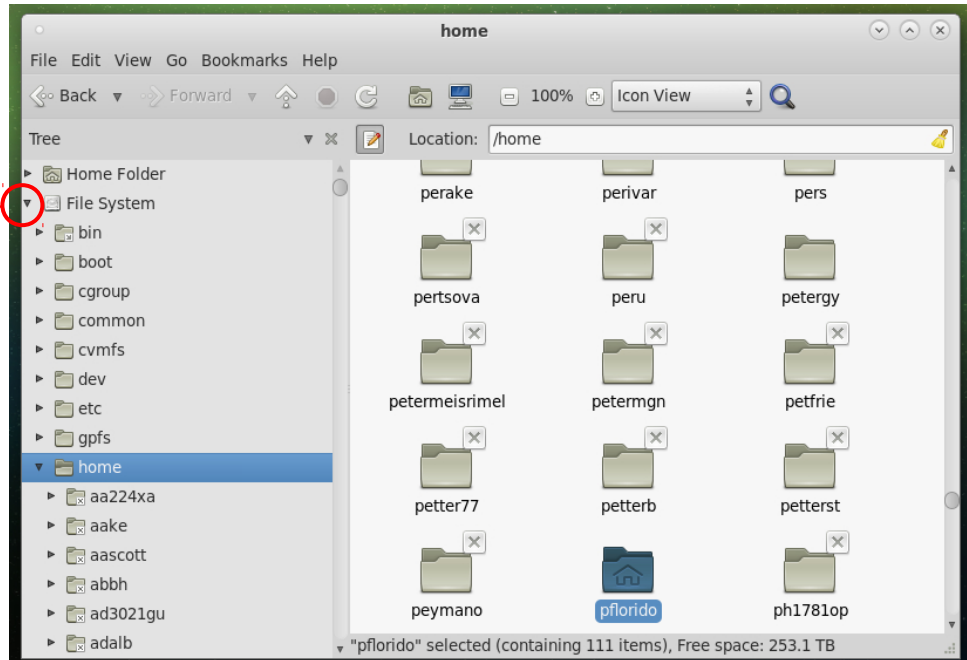


To create new folders, files and do other operations, **right click** on any white space between files in the file browser window or on the desktop. You can also open a terminal in the current filebrowser path.

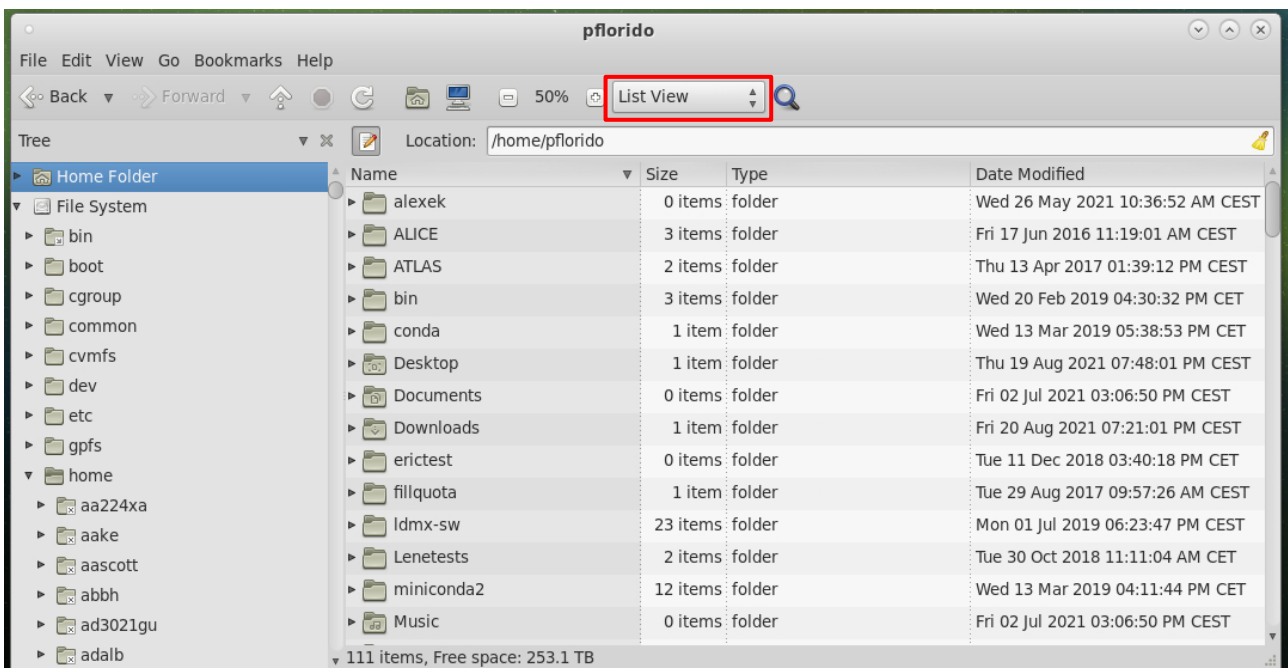


4. Tools on the HPC desktop

You can have a look at the whole frontend filesystem by clicking on such word in the left panel. For example in the picture below I expanded the filesystem *tree* by clicking on the small triangle on the left to navigate until the place where all users home folders are. You can see my home folder highlighted among the others. You should see yours if you do the same. The X icon on other people's folders indicate that you do not own that folder and in some cases you cannot browse them.

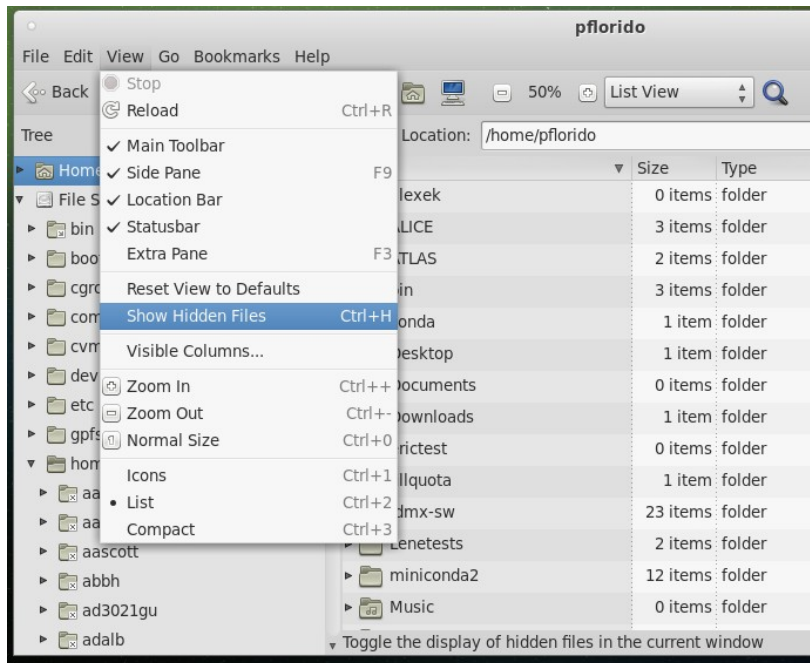


It may be useful to switch into **list view**, to order the files by size or date. To do so click on the dropdown box marked below and select the view:




4.Tools on the HPC desktop

Browsing through the **View** menu you can also make the hidden files visible.




4.3.1 Shortcuts

You can select one or multiple files by left clicking on the file or on the white space between files. A marker will appear to show you the selection.

You can move files around by keeping the left button  pressed after selecting them and moving your mouse around.

Del (Delete): Move selected file to trashcan

 **Del** (Shift+Delete): Remove selected file forever. It will not go to the trashcan.

Ctrl **n** : Open a new filebrowser window. By default it opens in your home folder.

4.4 A simple text editor: Pluma

We decided to use Pluma as the main text editor for this course. A text editor is a tool to modify text files, or files that contain text. This is basically all that you will do in this course, edit text files. So I suggest you learn very well the basics of how this is done.

There is no icon to open pluma. The fastest is to open it manually from a terminal.

1. Open a terminal (see 4.2)

2. type:

pluma&

```
[pflorido@aurora-rviz01 ~]$ pluma&
```

NOTE: the ampersand & symbol is important!

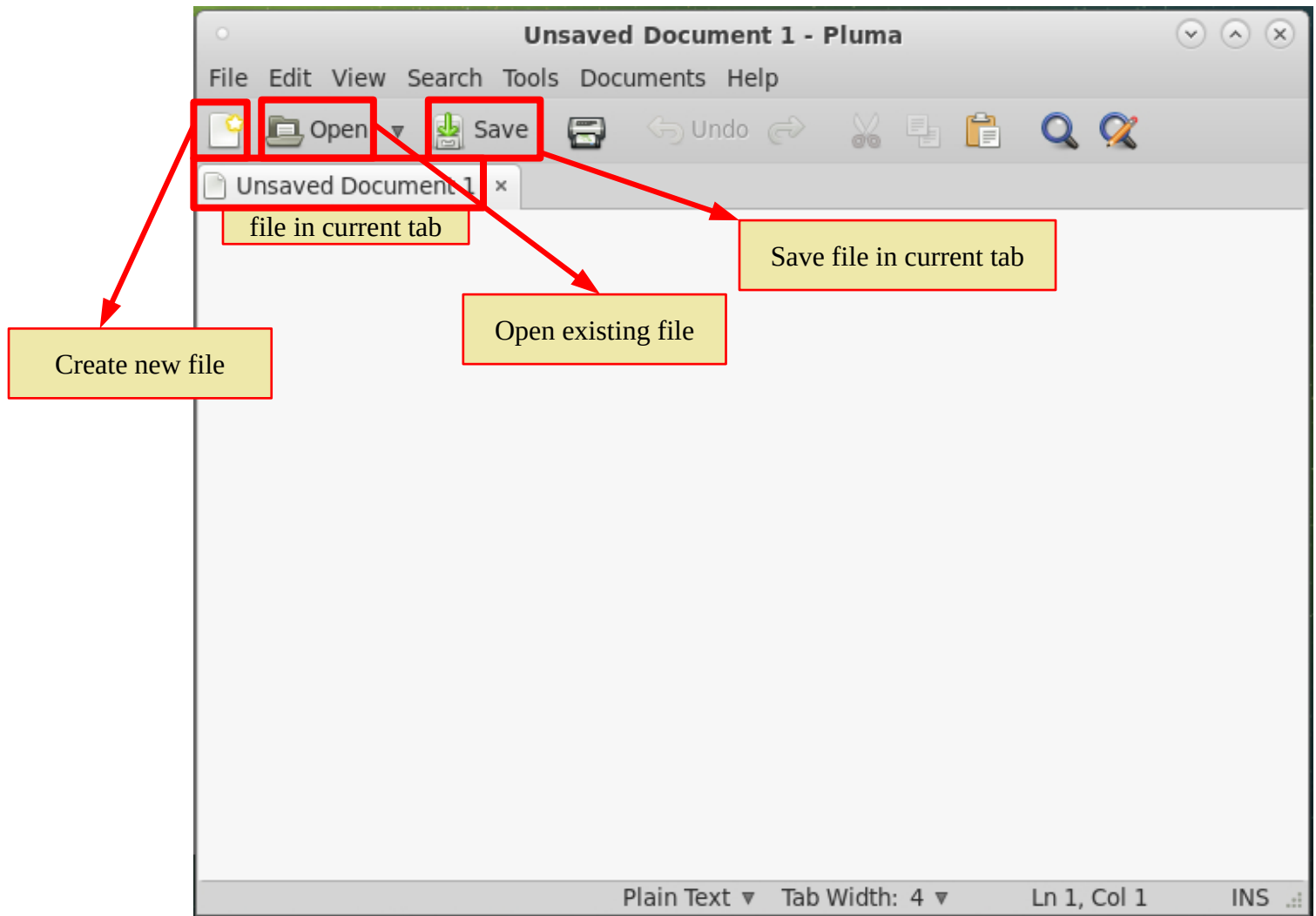
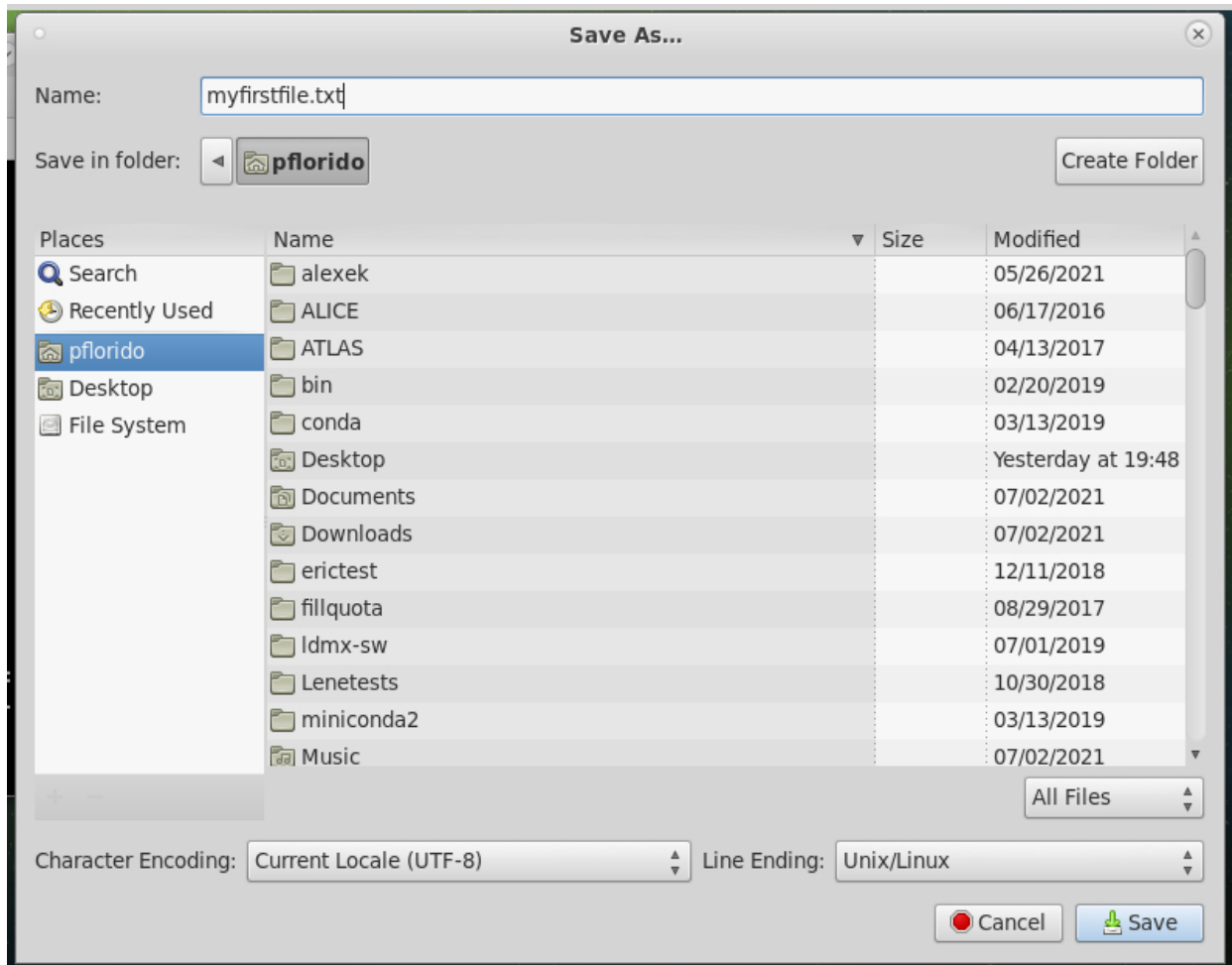


Illustration 4: The Pluma editor main window

4.4.1 Editing and saving your first file

1. Open `pluma` (see above)
2. Type in some text in the white area
3. Click on `Save`. The program will open a file browser. The picture below is relative to my files, yours might look very different.



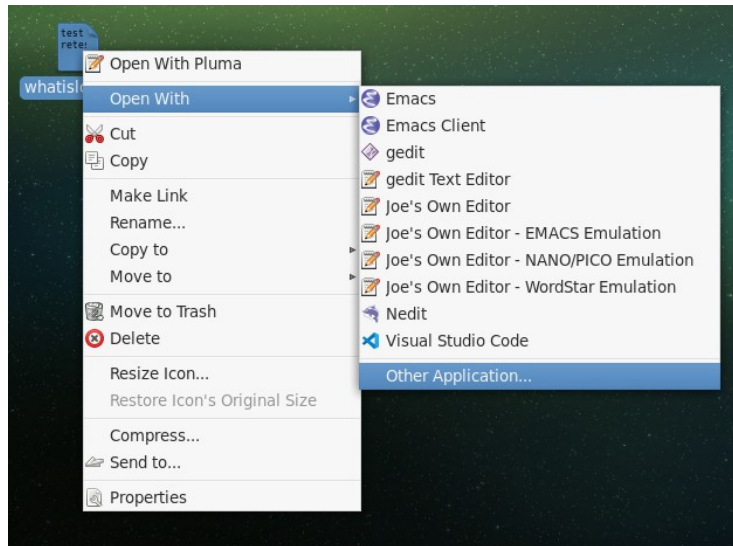
- 3.1. Select your home folder, in my case `pflorido`. You can identify it by a drawing of a house on the folder icon.
- 3.2. In the field *Name*, Type a filename, for example `myfirstfile.txt`. The *extension* `txt` indicates to a human reader that this should be a text file.
- 3.3. Notice the two dropdown menus *Character Encoding* and *Line Ending*. These are part of the *format* of the file. We will discuss them later in the course. Leave them as is.
- 3.4. Click on `Save`

Now you can use the file browser described in section 4.3 to find the file you just created.

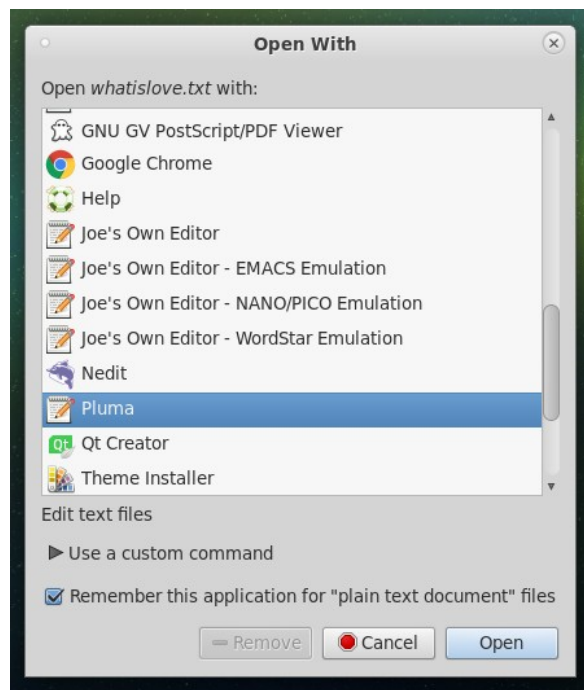
4.4.2 More on opening existing files with Pluma

One can open Pluma and click on the open button as shown in illustration 4. But there are other ways:

1) Graphical. One can right-click on a file that one wants to open and browse the available apps by clicking on *Open With* → *Other Application...*. For example here I am trying to open the *text file* *whatislove.txt*:



Then, select Pluma from the list of available applications. If you want to make Pluma the default application to open a file format when you double left click on it, tick also the checkbox “Remember this application for <file type here> files” as in the picture below.



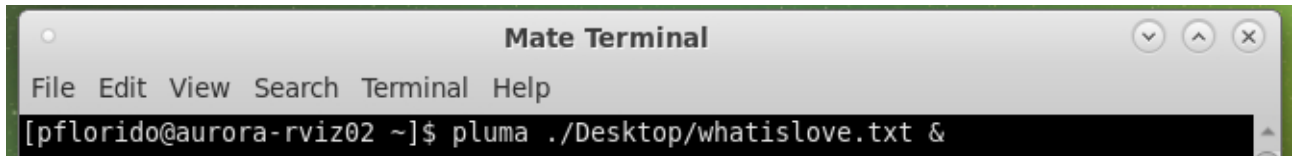
4.Tools on the HPC desktop

2) From the terminal. Open a terminal and write the command:

```
pluma [ ] <path/to/file/filename.ext> [ ] &
```

For example, to open the file `whatislove.txt` which is located on my Desktop, I ran from my *home folder*:

```
pluma ./Desktop/whatislove.txt &
```



4.4.3 Shortcuts

You can always select some text with the left mouse button and right click on it to see what operations you can do on such text.

Double left clicking on a word selects the word.

Triple left clicking selects a whole line.

- Ctrl** **n** Open a new file in a new tab
- Ctrl** **s** Save the file in the current tab

- Ctrl** **c** Copy
- Ctrl** **v** Paste
- Ctrl** **x** Cut

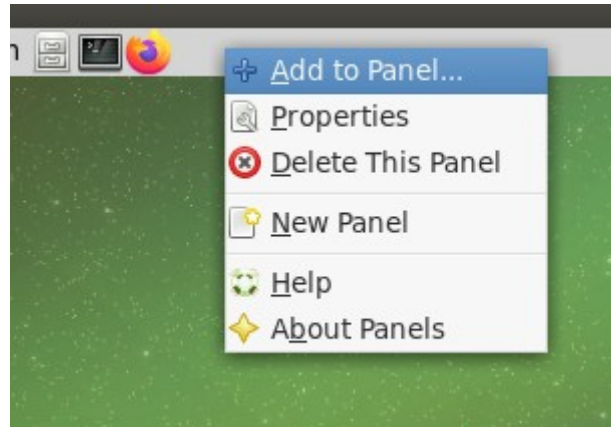
- Ctrl** **z** Undo (cancels the latest edit)
- Ctrl** **y** Redo (restores the latest undone edit)

4.4.4 Customizing Pluma for the course

In this section you will find instructions on how to configure the Pluma editor for the best coding experience.

4.4.4.1 Create a menu shortcut for Pluma

1) Right click on an empty space in the panel next to existing apps icons

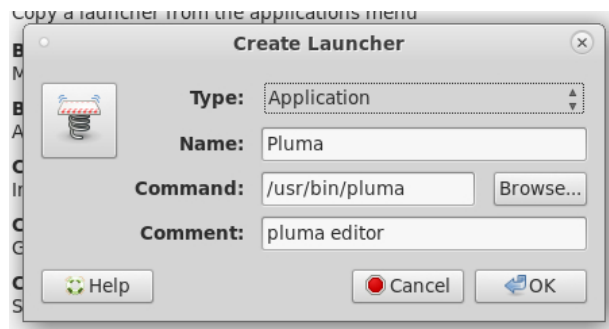


2) Select “Custom Application Launcher” and click OK




4. Tools on the HPC desktop

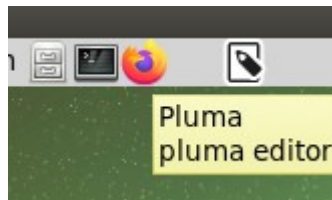
3) Fill the dialog with the following information as in the picture:



At this point you can click OK.

Optional: If you do not like the (ugly) default icon , you can change it later by right clicking on the app launcher icon you just created and click on the icon picture, it will open a file dialog to pick another image. I used the image at this path:
`/usr/share/icons/HighContrast/scalable/apps/accessories-text-editor.svg`

4) Now you can test the application launcher by clicking on the new icon.

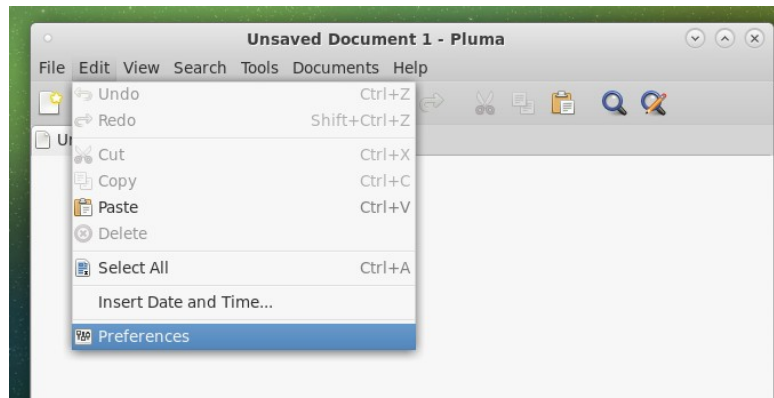


If when you click on it the Pluma editor does not come up, right click on the app launcher icon you just created and make sure the fields are exactly like in the picture at step 3).

4.4.4.2 Configure some pluma options that are useful for coding

There are a few useful feature that will help us coding so we should enable them.

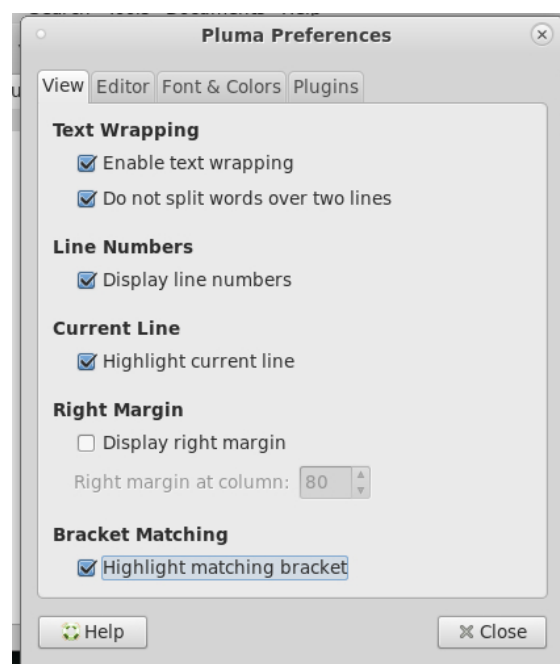
- 1) Open the Pluma editor (for example by clicking on the icon created at 4.4.3.1)
- 2) From the menu *Edit* select *Preferences*



3) I suggest you enable the following features:

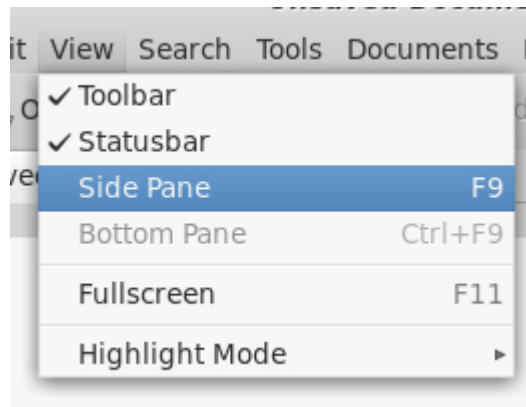
- text wrapping
- Do not split words over two lines
- Display line numbers
- Highlight current line
- Highlight matching bracket

To do so check the tickboxes as in the picture below and then click OK.



4. Tools on the HPC desktop

4) Optional: you can turn on the sidebar to browse open files or even files in the system without using an external filebrowser. You can enable and disable it upon you needs from the *View* menu or by pressing **F9**.



5. Command Line Quick Reference

Whenever you forget something about command line and commands you should browse this section. It should give you enough information to know what to do.


5.1 Command line syntax

This section describes how to type proper commands.

A command *syntax* usually looks like this:

```
command [ -o | --option ... ] [options] [parameter1 ... parameterN]
```

A *command* is always separated from its *options* and *parameters* by a *blank space* . Areas marked as above are *spaces*, one needs to press the spacebar key to separate the different *command parameters*. Spaces are usually **not visible**, I am showing them here for the reader to get used to them. They will not be shown in the rest of the course, you should make an effort to distinguish between a command and its options or parameters instead of blindly copying what you're told to.

To **execute** the command one should press the enter key: 

Square brackets [] and *triple dots ...* in the syntax above are usually **not** things one types when issuing the command but they're used to describe **optional elements** of the syntax.

Meanings:

[] : You may (or may not) include the content of the brackets [].

... : You may (or may not) type a list of elements separated by spaces

A *parameter* is usually the object on which the command will operate. Like an argument *x* of a function *f(x)*

An *option* and it is generally used to *modify the behavior* of the command.

An option may or may not have a parameter itself depending on the cases, i.e. `-o optionparam`

Here is an example based on the `ls` command:

```
ls [OPTION] ... [FILE] ...
```

the `ls` command `LiSts` files. Gives information about files and directories.

Example **parameter(s)**:

Name of one or more **FILES**, in the example below, two files (two parameters), the first file does not exist:

```
ls /home/pflorido/afile ~/.bashrc
ls: cannot access /home/pflorido/afile: No such file or directory
/home/pflorido/.bashrc
```

Example **option**:

Option without parameter: **-l** means "long listing", gives more detail about a file

```
ls -l /nfs/users/pflorido/afile ~/.bashrc
ls: cannot access /nfs/users/pflorido/afile: No such file or directory
-rw-r----- 1 pflorido hep 22 24 jan 2019 /nfs/users/pflorido/.bashrc
```

Example **option with parameter**: **--time-style=parameteroption**

changes the style with which the date is shown

```
ls -l --time-style=full-iso /home/pflorido ~/.bashrc
ls: cannot access /nfs/users/pflorido/afile: No such file or directory
-rw-r----- 1 pflorido hep 22 2019-01-24 18:41:05.586189098 +0100 /home/pflorido/.bashrc
```

5.2 Terminology and special characters

Terminology:

- A **string** is any sequence of characters or symbols.
- A variable **value** in a UNIX/Linux shell is anything prefixed by a \$ sign (sometimes referred as dollar, even if it's not the exact symbol)
Example: \$SHELL is the content (value) of the variable called SHELL , that contains the name of your default shell app.
- **standard input** is whatever one writes in the terminal using a keyboard.
- **standard output** is whatever one can see on the terminal.
- **standard error** is a channel where errors are shown. Usually it is redirected to the terminal *standard output* so that the user can see the error.

Special characters:

& : (ampersand) when placed at the end of a command, sends a command directly in the background

~ : shortcut, path to your home folder

Example:

```
ls ~
```

shows content of your home folder

\$: variable symbol

Example:

```
echo $SHELL
```

tells you the name of your default shell

| > < redirect command output. See Balazs lecture for details.

Example:

```
echo /etc/services | grep ftp > ~/extractfromservices.txt
```

The above:

1. **echo** prints the content of **/etc/services** on the terminal *standard output*
2. the pipe **|** captures all that should have gone to the *standard output* and passes it to the program on its right side
3. **grep** obtains from the pipe the content of **/etc/services** and searches for the string "ftp", prints the results of its findings on the terminal *standard output*
4. the right bracket **>** captures all the result of **grep** and instead of printing it on the terminal, writes them in a file called **extractfromservices.txt** in your home folder (**~**).

5.3 Useful commands

- **pwd** : **p**rint **w**orking **d**irectory, shows your home folder absolute path
- **ls** [**path**] : **l**ist directory contents at given **path**
- **cd** [**path**] : **c**hange **d**irectory to the given path, used to move around directories
- **wget** [**URL**] : **w**ww **g**et , **download** files from the internet at given URL
- **file** [**filename**] : shows **information** about file
Example: **file -i /etc/services**
- **head** [**filename**] : show the **first n lines** of a file
Example: **head -n10 /etc/services**
- **tail** [**filename**] : show the **last n lines** of a file
Example: **tail -n10 /etc/services**
- **less** [**filename**] : interactive **file reader**, exit by pressing "q", move around with arrows
Example: **less /etc/services**
- **touch** [**filename**] : change the time a filename was accessed. Can be used to create files.
Example: **touch ~/myfile**

5.Command Line Quick Reference

- **rm [filename]** : **remove** files
Example: `rm ~/myfile`
- **rmdir [path]** : **remove directory** at path, only when it's empty.
Example: `rmdir /emptydir`
- **exit** : exit a shell
- **echo [string]..[variable]** : **print** out a string or the content of a variable
Example: `echo $SHELL`
- **grep [pattern] [filename]** : **search** for the string pattern in filename and return the line that contains it
Example: `grep ftp /etc/services`
- **tar [options...][f tarball] [files...]** : **compress/extract** files into/from a tarball. the different actions depend on the options.
Examples:
 - Compress:
`tar cvf mytarball.tar file1 file2 file3` : compresses file1 file2 file3 into a mytarball.tar archive
 - Extract:
`tar xvf mytarball.tar` : extracts the contents of mytarball.tar into the current directory
- **clear** : cleanses up the terminal contents
- **reset** : restores the terminal in the default state. This is useful when the terminal behaves in weird ways due to inserting strange characters.
- **jobs** : list jobs in background.
- **fg** : brings the latest job that was in background to foreground.
- **bg** : send a currently suspended job in the background and keep running it.

A. Advanced topics

In this section there are a few things that we do not consider so important for the course but you might find interesting to try.

A.1. Sharing files live between your computer and Aurora using the ThinLinc client

DISCLAIMER: NOT RECOMMENDED. The preferred way to transfer data to a cluster is with a standalone program as explained in section 3.3.

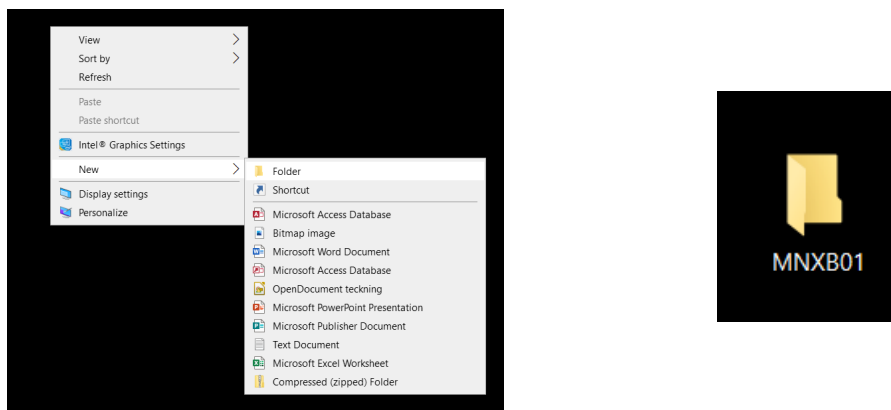
The method described here makes it easy to see selected folders on your desktop on the Aurora cluster. While this is a very nice and practical feature, **I do not recommend it**. It gives a false feeling that one can seamlessly modify files on your computer from Aurora. The problem is that this is extremely inefficient and error prone, that is, it happened to me several times that the HPC desktop could not save these files or had troubles synchronizing them between the two machines. Moreover, it is safe only if you use exactly the same computer to connect: if you use more than one computer I had several problems with unexpected issues, some described later. If you are using multiple device of need to transfer big files, You should definitely follow the instructions in section 3.3 instead.

However there is one case where it is very useful, and it is when you want to edit the files you want to send to Aurora from your own computer, maybe with your favorite editor. Since it is a useful feature and maybe it gets better in the future, I explain it here.

To setup folder sharing, logout from Aurora and close the ThinLinc app. This operation cannot be done while the app is running and it may not work if you have existing open sessions.

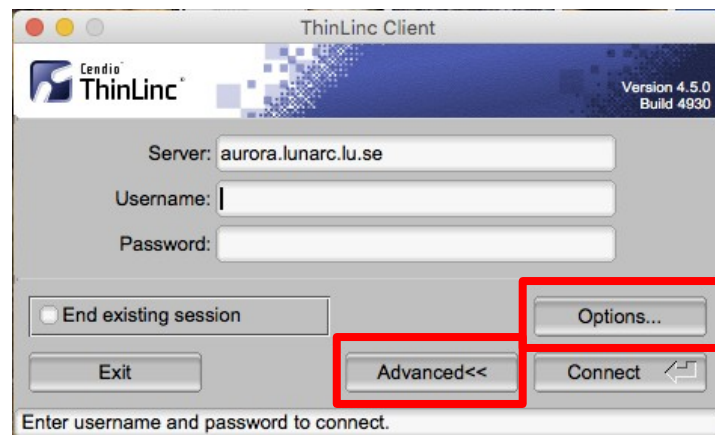
The instructions below are Windows 10 only, but the feature should work on both Mac and Linux.

1) Create a folder MNXB01 on the Desktop of your computer

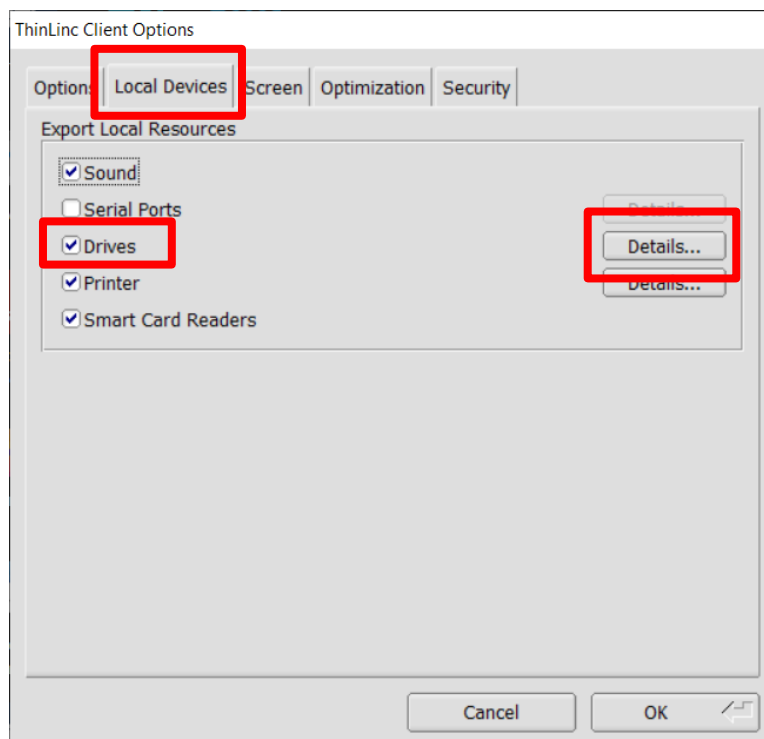


A.Advanced topics

2) Open the ThinLinc app and choose “Advanced”, then click on “Options”

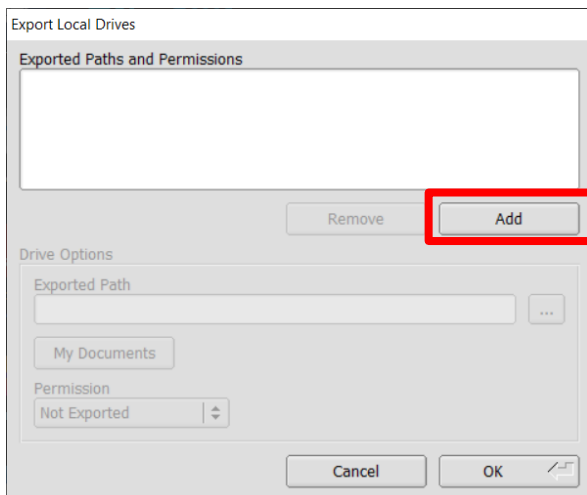


3) Navigate to the “Local Devices” tab. Tick “Drives” as shown in the picture below, and then click on “Details”

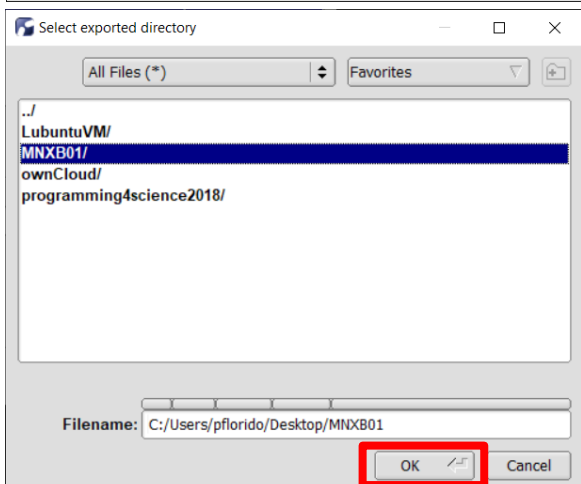
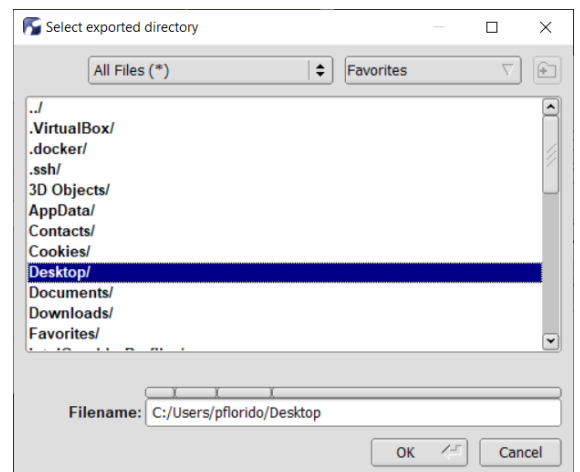
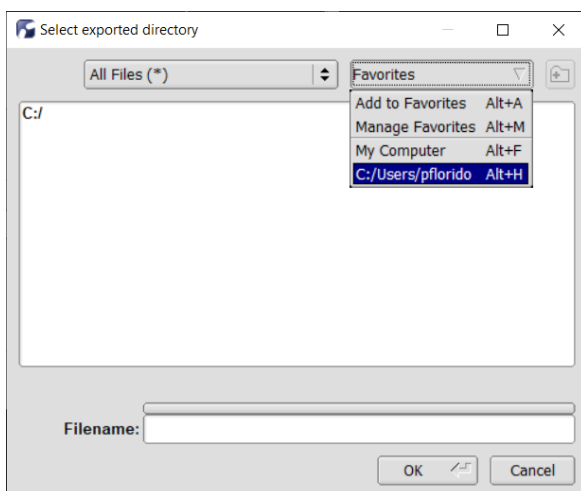


A.Advanced topics

4) There might be weird things in the details page. Usually a not well defined “d” folder which you may delete. In this example below I decided to remove all pre-configured shares and add only the MNXB01 folder. Click on “Add”

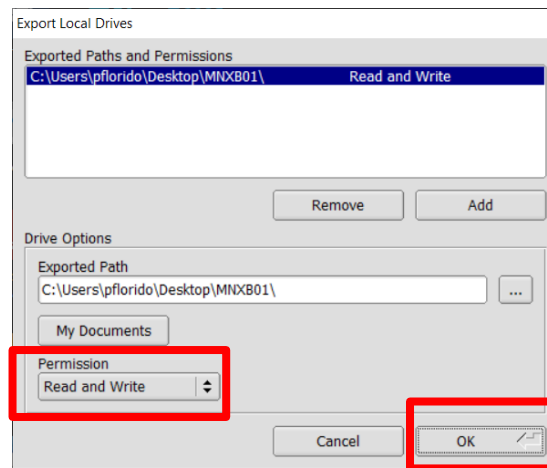


5) Select the MNXB01 folder by clicking on the [...] button and using the filebrowser to find the folder. Your windows Desktop is usually located in C:\Users\yourusername\Desktop. Once you found the folder, click OK.



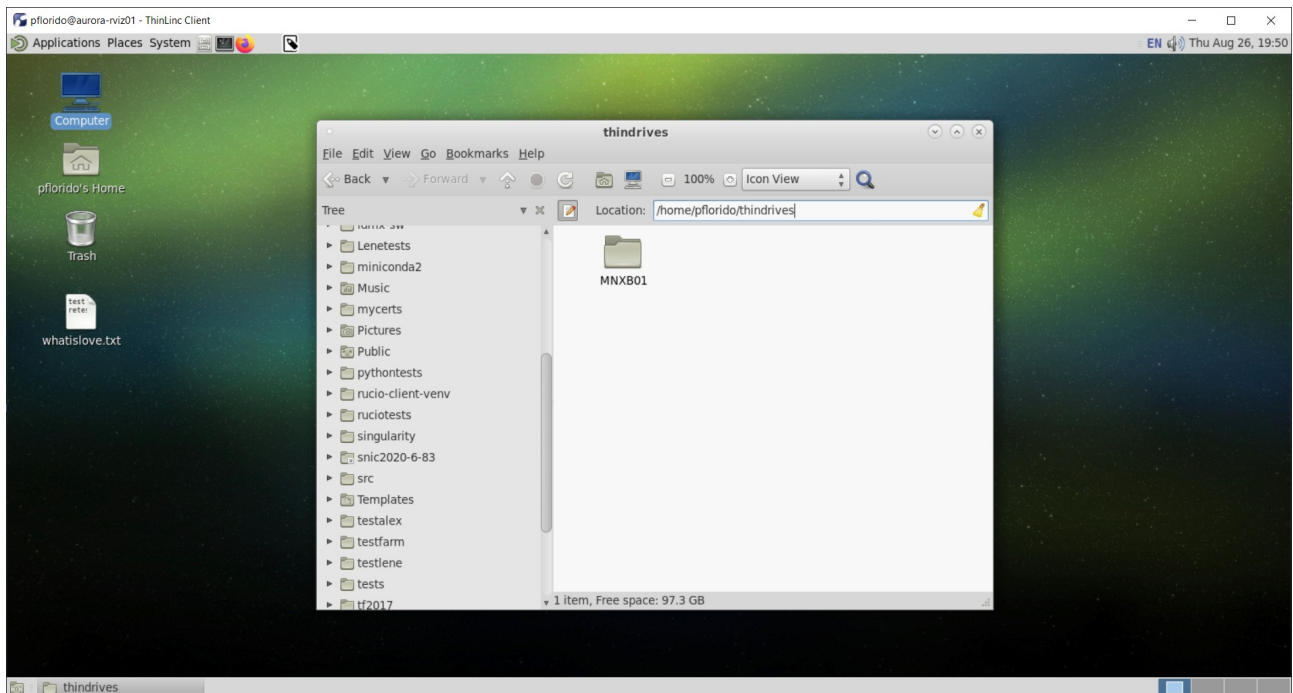
A.Advanced topics

6) Select “Read and Write” as permission if you want to be able to modify the files in Aurora. Click OK.

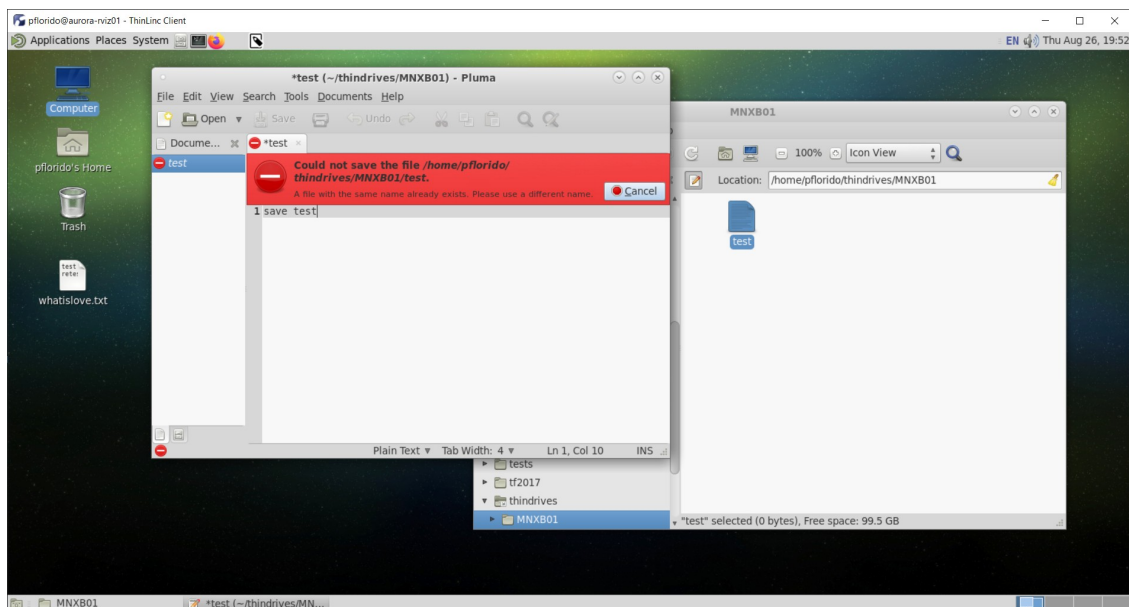


A.Advanced topics

7) Login to Aurora as usual. Once in the HPC desktop, you should be able to see your shared folders in your home in the special folder **thindrives**, at the path `~/thindrives/` :



Warning: Pluma and some other applications does not seem to work well with this feature. After editing a file, when trying to save, I always got this:



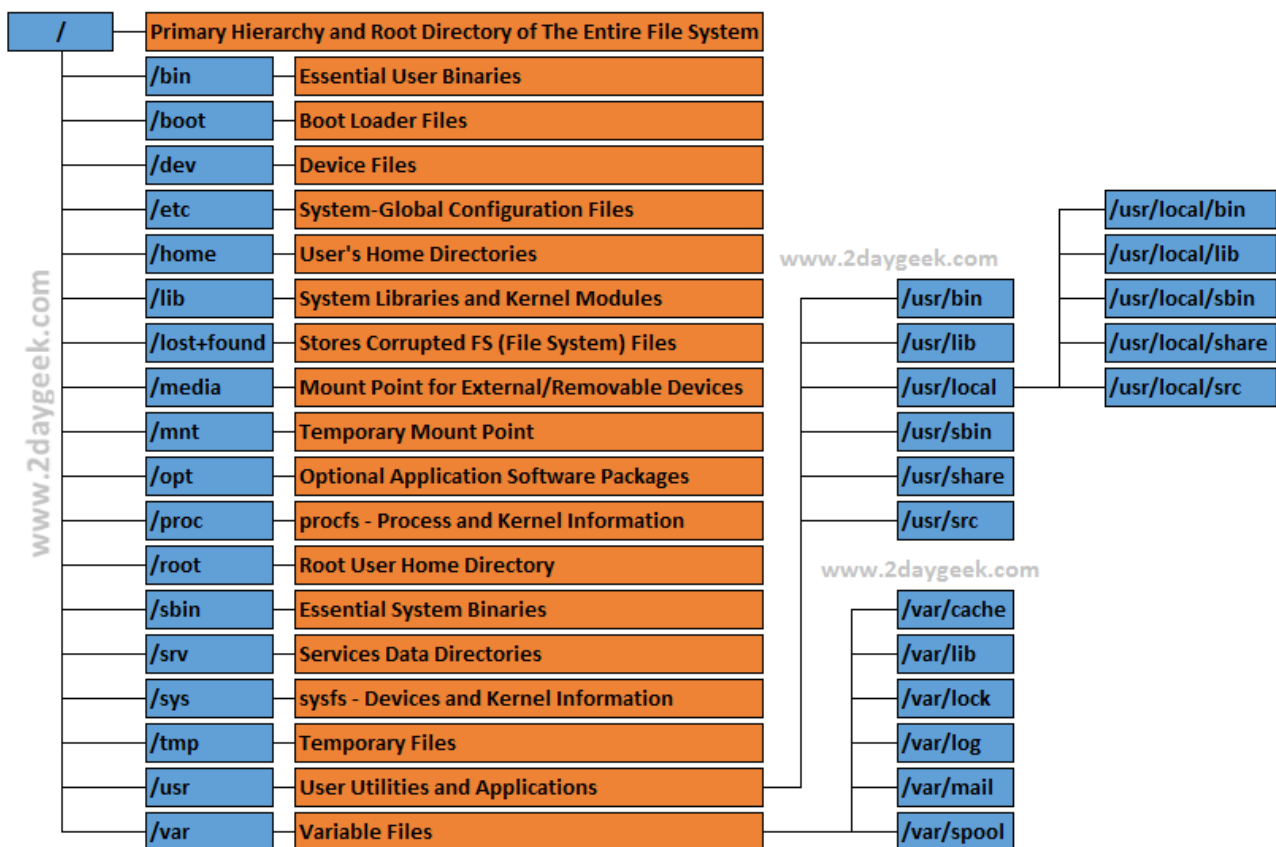
However, editing/saving with a command line editor (Vim, Emacs...) always worked.

A.2. The Linux Filesystem

The linux filesystem is organized as a tree of *folders* or *directories*. The so-called "root" of the tree (that has nothing to do with the root user!) is identified by a slash symbol /

Directories, or folders, containing files can be described as "paths" on the tree. Every folder is separated by a slash symbol.

For your convenience, a picture with the different folders and their common purposes is provided below (source: [tdgLFS]):



Your **home folder** is where your personal files are, normally, this is located at the path /home/yourusername or /home/yourusername/

The trailing slash / at the end of the path it is sometimes optional, but it's always better to add it to show you are referring to a folder.

These paths are called **absolute** because they show a **direct way to reach a folder** starting from the root /.

At any point while in the terminal or in the filebrowser your user will be in a **current directory**.

The **current** directory is usually identified by a **single dot**: . or ./

The **parent** directory in the tree hierarchy is identified by **double dots**: .. or ../

A.Advanced topics

For example is my user current folder is my home folder with absolute path `/home/pflorido`:

- `./` points at the absolute path `/home/pflorido`
- `../` points at the absolute path `/home`

Relative paths, discussed in Tutorial 2, are those who make use of the dot and double dots. Sometimes these symbols are implicit. For example, if my current folder is `/home/pflorido/`, my Desktop folder with absolute path `/home/pflorido/Desktop/` can be found at `./Desktop/` or simply `Desktop/` (with no preceding dots and slashes)

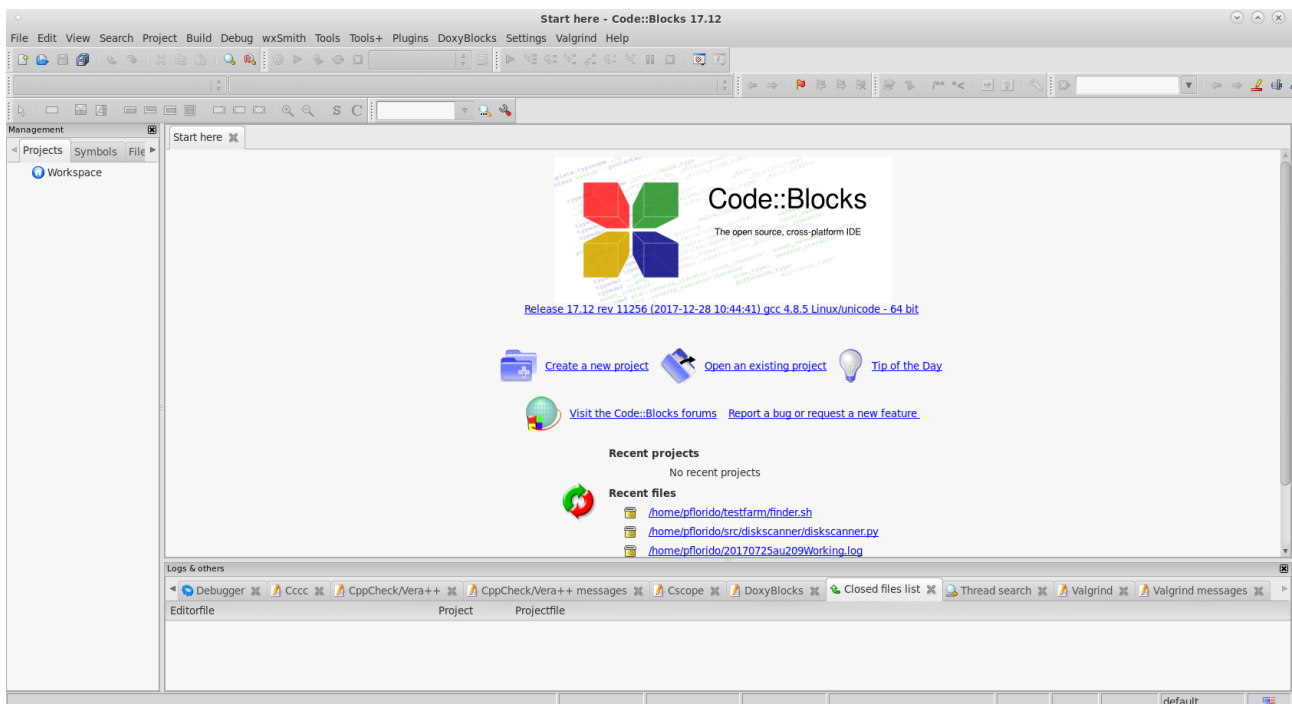
A.3. An IDE: codeblocks

As you progress with your programming experience, you will find that a simple text editor is too limited. You need more powerful tools to maximize your coding productivity. These are usually called IDE, Integrated Development Environment, which are basically text editors with support for external coding tools.

In this course we will briefly suggest an IDE called codeblocks that is a reliable open source tool. I will mainly use it in this course to show you some details about file format, but it is not the purpose of the course to teach you how to use an IDE.

You can open codeblocks by running it from the terminal, with the command

```
codeblocks&
```



Instead of reasoning in terms of files, IDEs usually think in terms of Projects and force you to do a lot of preliminary work before you start coding. This is usually to keep things tidy, but in my personal experience this always leads to complexity that is not very well justified, and makes

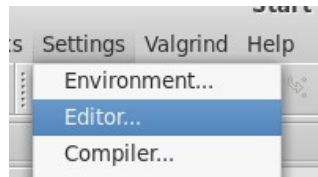
A.Advanced topics

changing IDE a painful process as they are all different and their concept of project is not compatible.

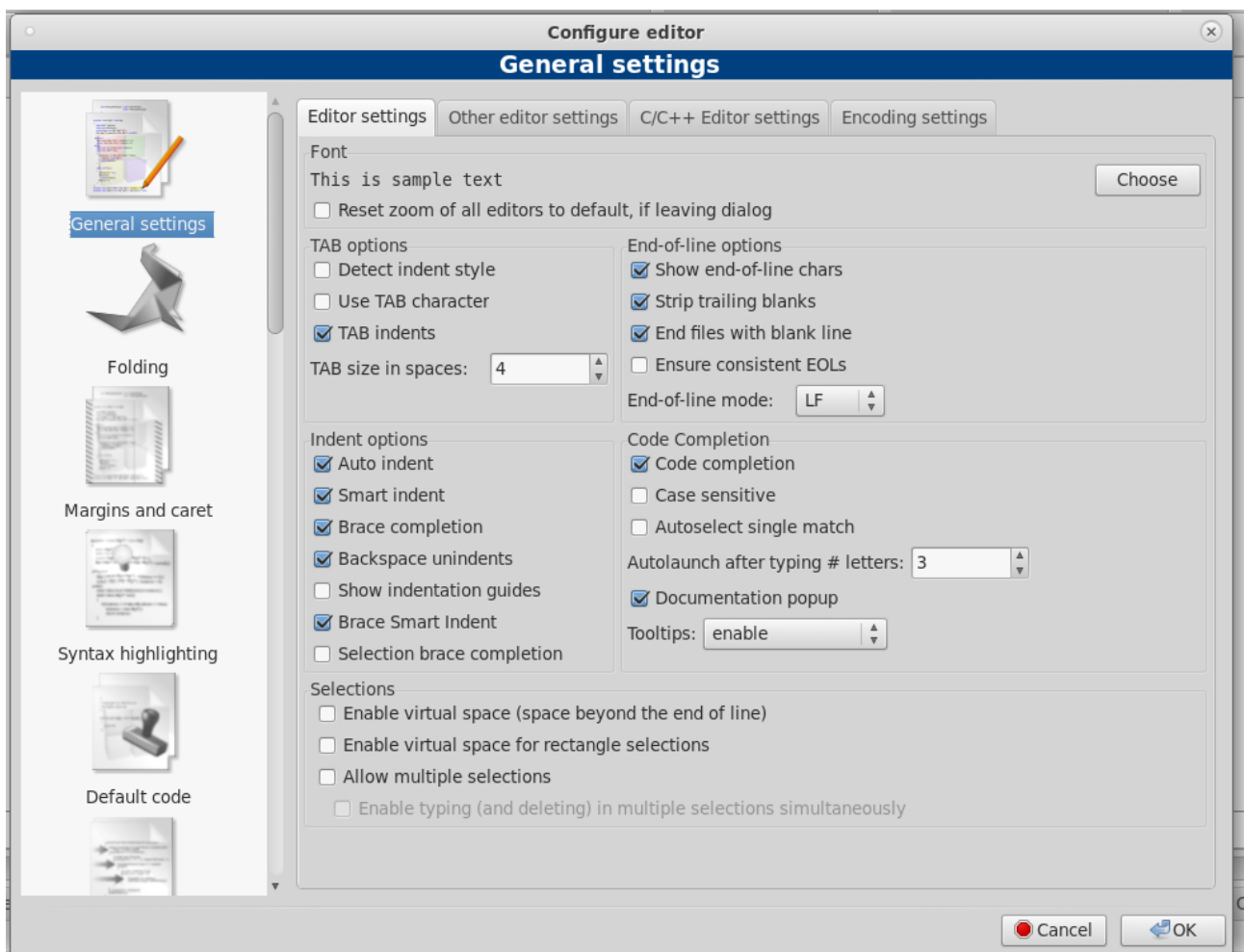
Nevertheless you can still use them as an advanced text editor.

A.3.1. Visualizing hidden characters

To visualize hidden control characters such as spaces, tabs and carriage return / line feeds, find the Settings menu and select Editor...

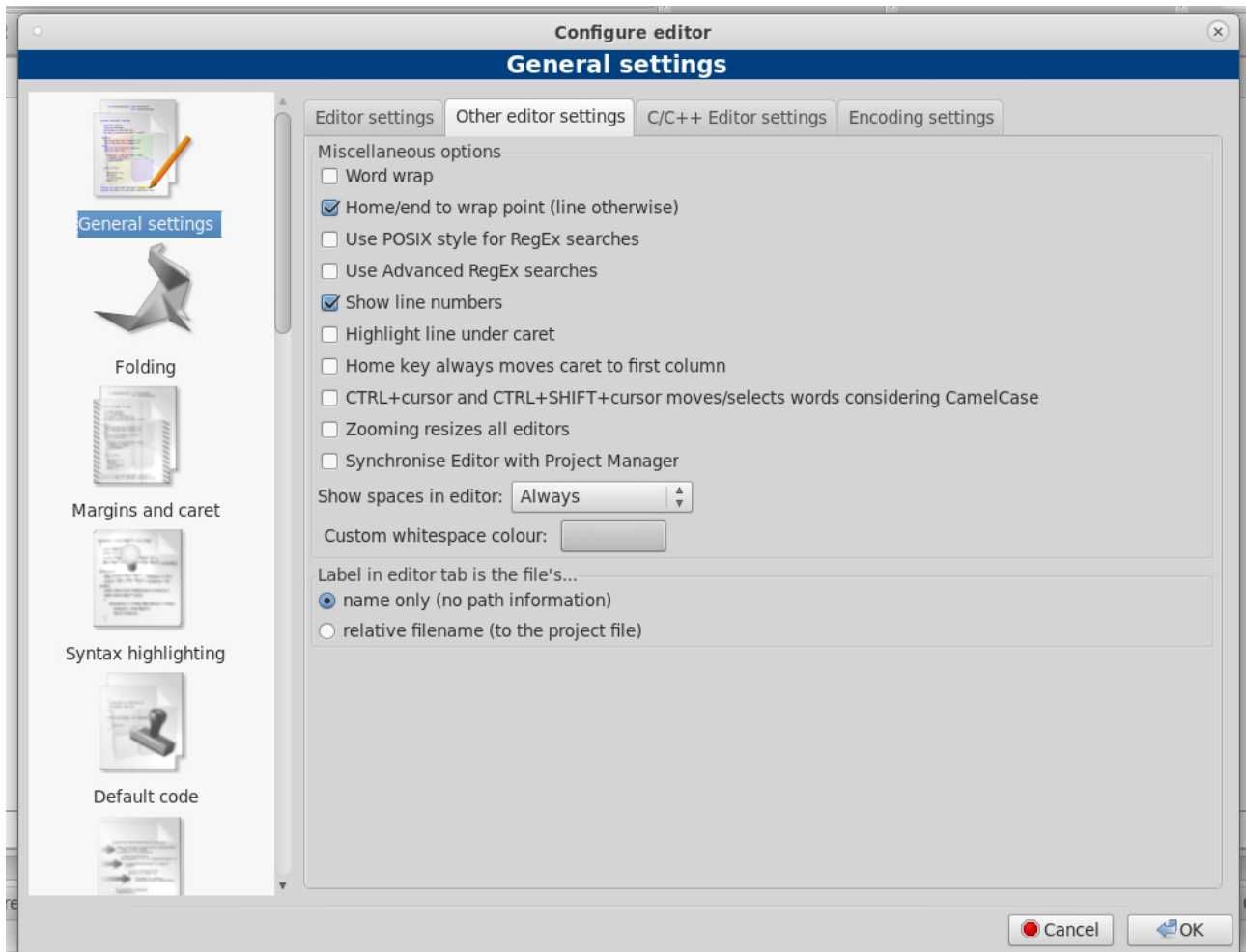


In the *Editor settings* tab and change the settings according to the pictures below. The options we're interested in are *Show end-of-line-chars*:



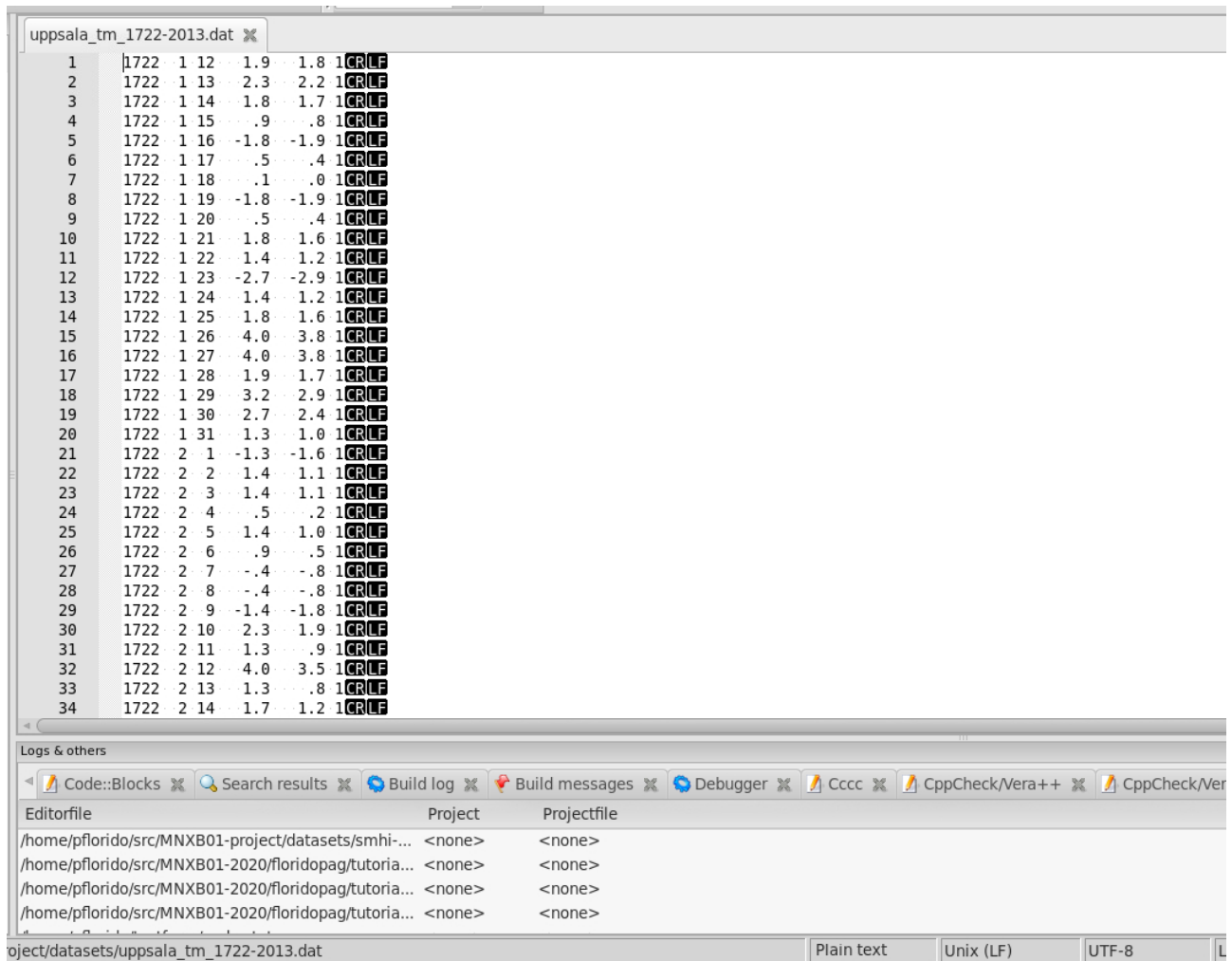
A.Advanced topics

and in the Other editor settings tab, *Show spaces in editor: Always*



A.Advanced topics

The result is shown in this example file of data that we will use for the project:



```
uppsala_tm_1722-2013.dat
1 1722 1 12 -1.9 -1.8 1CRLF
2 1722 1 13 -2.3 -2.2 1CRLF
3 1722 1 14 -1.8 -1.7 1CRLF
4 1722 1 15 -0.9 -0.8 1CRLF
5 1722 1 16 -1.8 -1.9 1CRLF
6 1722 1 17 -0.5 -0.4 1CRLF
7 1722 1 18 -0.1 -0.0 1CRLF
8 1722 1 19 -1.8 -1.9 1CRLF
9 1722 1 20 -0.5 -0.4 1CRLF
10 1722 1 21 -1.8 -1.6 1CRLF
11 1722 1 22 -1.4 -1.2 1CRLF
12 1722 1 23 -2.7 -2.9 1CRLF
13 1722 1 24 -1.4 -1.2 1CRLF
14 1722 1 25 -1.8 -1.6 1CRLF
15 1722 1 26 -4.0 -3.8 1CRLF
16 1722 1 27 -4.0 -3.8 1CRLF
17 1722 1 28 -1.9 -1.7 1CRLF
18 1722 1 29 -3.2 -2.9 1CRLF
19 1722 1 30 -2.7 -2.4 1CRLF
20 1722 1 31 -1.3 -1.0 1CRLF
21 1722 2 1 -1.3 -1.6 1CRLF
22 1722 2 2 -1.4 -1.1 1CRLF
23 1722 2 3 -1.4 -1.1 1CRLF
24 1722 2 4 -0.5 -0.2 1CRLF
25 1722 2 5 -1.4 -1.0 1CRLF
26 1722 2 6 -0.9 -0.5 1CRLF
27 1722 2 7 -0.4 -0.8 1CRLF
28 1722 2 8 -0.4 -0.8 1CRLF
29 1722 2 9 -1.4 -1.8 1CRLF
30 1722 2 10 -2.3 -1.9 1CRLF
31 1722 2 11 -1.3 -0.9 1CRLF
32 1722 2 12 -4.0 -3.5 1CRLF
33 1722 2 13 -1.3 -0.8 1CRLF
34 1722 2 14 -1.7 -1.2 1CRLF
```

An interesting thing is that codeblocks “lies” about the actual format of the file. This is because in the options *Encoding settings* every file we open is forced to be UTF-8 and Unix LF. This is one of the aids an IDE provides to a programmer to keep their code consistent.

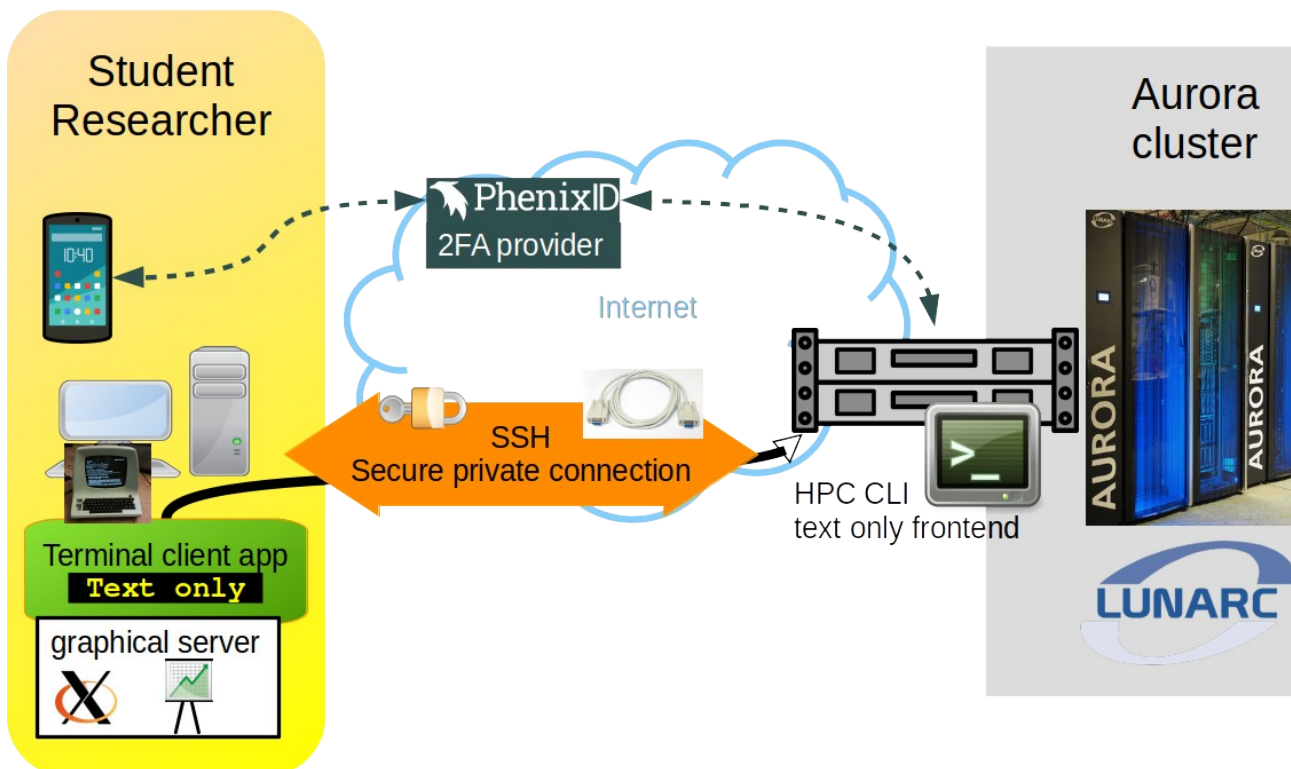
We can discover the actual format of the file using the `file` command:

```
[pflorido@aurora-rviz01 datasets]$ file uppsala_tm_1722-2013_description.txt
uppsala_tm_1722-2013_description.txt: ISO-8859 text, with CRLF line terminators

[pflorido@aurora-rviz01 datasets]$ file -i uppsala_tm_1722-2013_description.txt
uppsala_tm_1722-2013_description.txt: text/plain; charset=iso-8859-1
```


A.4. Text-only connection: Using a terminal emulator (Optional)

Terminal emulators are actually the best way to connect to a cluster. As you proceed in your career you will find that it is a much more efficient way to use a cluster to just login with a text interface, because most of what you will want to do there is just to start your calculations based on parameters.



Here I list some popular terminal emulators for most common Operating Systems.

A.4.1. On Windows

Do NOT search on google. Follow the links shown below to avoid viruses and threats.

- Download and install the following from these sources:
 - PuTTY (terminal emulator):
<https://the.earth.li/~sgtatham/putty/latest/w64/putty-64bit-0.77-installer.msi>
 - More info can be found at this link:
https://www.hep.lu.se/staff/paganelli/doku.php/it_tips:xwindowswithmswindows

- You can connect via putty by writing aurora.lunarc.lu.se as hostname in the configuration. Read the manual here: <https://the.earth.li/~sgtatham/putty/0.77/html/doc/>

A.4.2. On Linux or MacOS

Open any terminal emulator: which one you have depends on the *Window Manager* (GNOME, KDE, LightDM, XFCE) that supports some virtual terminal apps. Some examples are: Gnome-terminal, xterm, konsole, MacOSX Terminal...

To connect via ssh you will need the **ssh** command line client. Usually you need to install the **openssh command line client ssh**. It is actually included by default so you usually do not need to install anything.

For those of you that want to know what software packages are required, in most cases it is sufficient to have the **openssh-client** package installed. Package names may vary so I will not provide an extensive list.

You can connect to Aurora by writing this command in the terminal:

```
ssh <yourusername>@aurora.lunarc.lu.se
```

```
Example: ssh florido@aurora.lunarc.lu.se
```

A.5. Command line file transfers

You can transfer files via command line as well. There are mainly two commands that transfer files via SSH: **scp** (non interactive) and **sftp** (interactive). I will however not document them here for the moment. But you can read the manpages by issuing **man scp** or **man sftp** on the terminal.

A.6. SSH key pairs

SSH keys are an implementation of PKI (Public Key Infrastructure). You can read about the generic concept in Lecture 4. In this section I will just describe how this is done in SSH.

Before we go into the topic, a clarification: SSH communication encryption does NOT use SSH keys. These are used only for authentication. The Secure in SSH is based on Symmetric Encryption, both the client and the server decide a secret that they share during the whole connection. It does not depend on the identity of the user or the server.

SSH key pairs are based on well established encryption algorithms to generate a **private** and a **public** key (Asymmetric Cryptography) to identify a *security principal*, be it a person (user) or a computer (server).

- The private key **should never be disclosed**, not even to those who you trust;
- The public key is **for everyone to read**, in fact, you can distribute it everywhere you like to whomever you want; it can even end up in malicious hands.

The security concept behind this two items is that to do some action you need both. A system that supports PKI uses the public key to encrypt (= mix up so that it makes no sense to a reader) the content to send to the owner of the private key. The mathematics behind the encryption scheme is such that only the person who owns the private key can decrypt (= extract or interpret) the content of a message encrypted with their public key.

Mathematically, private and public keys are values obtained from an algorithm based on the multiplication of prime numbers. The result of the algorithm is such that it is extremely hard to find those two numbers that would allow every encrypted message to be decrypted. By hard we mean that even the most powerful computer on earth may take million years to come up with a solution.

Of course as the computing power increases, this complexity reduces a bit. Nowadays the prime numbers one multiplies must be big enough so that no hacker can break the algorithm.

In practice, these values are encoded (=represented) as text files, sequences of characters that represent the mathematically calculated values. There are many algorithms that one can use to obtain such values. One of the most common is the RSA algorithm.

As you might imagine these encoded values look like pure garbage to us. This below is an example of generated private and public key pair:

```
-----BEGIN RSA PRIVATE KEY-----
```

[illegible]

A.Advanced topics

```
AHV1svAwP3LUdaBzN7D1rVpZzQ2Roaw09aecBVixM2orJptMcdmJpDqBuur50+rJRR2t2NNAj50tqP47tnZLv+e2ksyB+xbF7uxIaId22tmKgb9UD3HToJfzD1pD0kSV8cX1sV+xo+yr1lw/
jNnsPskxSj1l1a2vfJsmPztn4CL8lmpVdxoY28ftDX11Ls r1T4HFT3A5CcAYx6zLqg6XyShZNsC/WiHb/z7E249AychHAJ3GyvN0Ycy3s8GmkXohV9vKy5Gn0H4uhy12dkoTDnuq6thx/ArX81
```

-----END RSA PRIVATE KEY-----

Public key:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQACQ6jE0WUthp4/pGM2D9Zdx9xC8ChzDuA27kGGu/H7IE0ZUGjESdft3YpKnHzRjcwYHrNUqheNjQZj1lc+zhGVsbavjxQ3fuLDiuePh9+TY80b7f9YukKZH+TI8mKQcckPy45y4nD2yHkgqIH0G9FA2xU0LWtNIU/
D4Q2UDaUeSN0kobVHLKwX1Lo3qX7N7j/
dJ4PqvcFmU0JYkNE9vD0a90LmS8P0gJkmM2g3Kb9pUGLprLs6Xw0EHLZK6Ve6h+h4CU06is1AJzUyq0bh3kNRBbmHVZM1c4qb051vghguDowb1bEoEmfFDJ9Z11ngb0gASV96sK31Hho0UKRPoaRu0LVsueQ1r3unZ3Wz0D/Bj/
2C8dmwj3oC0NZD26Cz0J3K7gHxTA9/2KDJV1/iIK0TT70yfvrr6Zow2RUKsYPE/r+Q0Gv8UNbXowtLj0mE+2cKs9XfshNpMI3556v0q5xGw004ehI1e8g71V25QqslXsrSaImWJ3nb/
1l+V+nQUuSQK0D5zAMVct1Xk7+4pPm8k8ix05f1VCg11pqA1K9tjUg4XL2ieU36Q57bY3YgHmyFV4ELs04mgt5owpoDcxvUffW2Dx4e9KKA0gc0TVbH2xv9W0YXrcBbNHNQ8jjVWncMxSIIAmeXpqANs5o8j+CmMbIYbpUSiw== pflorido@atariXL
```

I think you can see by yourself that it just does not make sense to share or remember any of the above. For this reason, the public key that is supposed to be shared with others is usually **hashed** using an hashing algorithm. Hash functions map a certain set of values into a numerable, restricted set of symbols or values. In the case of SSH, the public key is hashed using some hashing algorithms such as MD5 or SHA to make them shorter. You can think of a hash as some kind of summary.

For example, the public key above has the following hash:

SHA256:ApGaiVK0bpDCJhM6puAHLzhgIGg+keMAN3kusjNbLo8

The hash is also called **fingerprint**. Without going into details, 256 determines the complexity of decoding the hash. Higher numbers are harder to decode. They also make the resulting hash longer.

These complex hashes have also another property, that is, they **identify in an almost unique manner** the hashed value. You can think of them as an identity card for the public key: if someone knows your public key and wants to verify that it is yours, they can ask you the hash and calculate the SHA256 over your public key, if they match they can be sure it's you.

In the digital certificate world we have a third party, a Certification Authority, that validates our trust, if you remember what is described in section 2.1 or in Lecture 4.

SSH key based authentication has no intermediary. The trust must be established outside the computer world by some shared knowledge between two parties. For you to trust a server, you need to accept their public key hash as you did in section 3.2 when logging in to Lunarc.

For the server to trust your public key the thing is a bit more complicated. It will not be happy with just a hash. You need to copy the whole public key over to the server, usually in your home inside a file `~/.ssh/authorized_keys`

In SSH key pair based authentication,

- 1) You trust the server SSH key fingerprint;
- 2) The server has a copy of your public key installed in your home folder;
- 3) When connecting, instead of sending username and password, the client sends your key fingerprint;
- 4) The server calculates the hash of your public key and checks that it matches your fingerprint;
- 5) If the calculated hash does not match, the connection tries other means of authentication (such as other keys or username and password) otherwise it continues, sending you a message encrypted with your public key (asymmetric cryptography)

6) If the client has your private key, it can decrypt the message, and then perform an algorithm that I will not describe here to complete the authentication.

SSH keys can have an unlock password. **I strongly recommend you to use a password for your keys.** A key without a password is like your bankID without pin code, or you id card without a picture. If someone finds it, they can pretend it's you.

The good thing about SSH keys is that they are managed by a so called *SSH Agent*. It's a small piece of software that remembers what key you used for which service, so that you only have to write your password once during a session. After that the agent will remember and you can login without a password. So it's a way to speedup or automate authentication.

A.6.1. Using SSH key pairs instead of username and password

The instructions below are Linux/macOS only. This guide will not cover Windows.

To use an SSH key pair instead of username and password, you need to do the following **on your laptop**:

1. Generate a private/public keypair **with password** and strong encryption:

```
ssh-keygen -b 4096 -f ~/.ssh/myid_rsa
```

Result:

```
Generating public/private rsa key pair.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /nfs/users/floridop/.ssh/myid_rsa.  
Your public key has been saved in /nfs/users/floridop/.ssh/myid_rsa.pub.  
The key fingerprint is:  
2d:1d:94:b9:71:35:59:f8:79:26:92:b5:a3:f5:d4:e3 pflorido@tjatte.hep.lu.se  
The key's randomart image is:  
+--[ RSA 4096 ]-----+  
|           .o .o+. |  
|          .+ . +. |  
|         .+ o oo |  
|        o..o =o* |  
|       S o  +.*o |  
|      .   . E. |  
|               |  
|               |  
+-----+
```

- ## 2. Make sure the permissions are correct:

```
chmod 600 ~/.ssh/myid_rsa;
chmod 644 ~/.ssh/myid_rsa.pub;
```

```
# check permissions
ls -ltrah ~/.ssh/myid_rsa*
```

A.Advanced topics

```
-rw----- 1 pflorido hep 3,3K maj  3 13:59
/nfs/users/floridop/.ssh/myid_rsa
-rw-r--r-- 1 pflorido hep 751 maj  3 13:59
/nfs/users/floridop/.ssh/myid_rsa.pub
```

3. Copy the key to the target server, say Aurora. This will add your public key to `~/.ssh/authorized_keys` on Aurora.

```
ssh-copy-id -i ~/.ssh/myid_rsa pflorido@aurora.lunarc.lu.se
```

Result:

```
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to
filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
Password:
```

Number of key(s) added: 1

Now try logging into the machine, with: `"ssh 'pflorido@aurora.lunarc.lu.se'"`
and check to make sure that only the key(s) you wanted were added.

4. Add the key to the agent keyring:

```
ssh-add ~/.ssh/myid_rsa
```

Result:

```
Enter passphrase for /nfs/users/floridop/.ssh/myid_rsa:
Identity added: /nfs/users/floridop/.ssh/myid_rsa
(/nfs/users/floridop/.ssh/myid_rsa)
```

5. Try to login to the server **using the identity created**:

```
ssh -i ~/.ssh/myid_rsa.pub pflorido@aurora.lunarc.lu.se
```

You will now be asked the OTP as usual.

B. Tips for installing your own Linux

If you have Windows or MacOSX and you want to try Linux at home, I suggest you just follow the previous years course slides.

The slides discuss both installing using virtualization or bare metal installations:

<https://www.hep.lu.se/courses/MNXB01/2019/MNXB01-tutorial-1.pdf>

If you have Windows 10, Microsoft created a partnership with Ubuntu to be able to use the Linux command line interface in Windows. You can try the following links for some hybrid approach:

Install Ubuntu bash terminal on windows 10

<https://ubuntu.com/tutorials/ubuntu-on-windows#1-overview>

The above guide does not cover the X server (the graphical interface). You can find some tips on my support pages:

https://www.hep.lu.se/staff/paganelli/doku.php/it_tips:xwindowswithmswindows

C. Glossary of terms

word	meaning	synonyms	related topics
2FA	Two Factor Authentication is an authentication scheme where the participants need to share at least two authentication credentials	Two Factor Authentication	
assembler	An app that can convert assembly language into machine code.		
assembly	An intermediate, low level representation of machine language that is more human readable		
authentication	A process with which two or more parties identify each other, based on some form of trust, by exchanging authentication credentials.		
authentication credential	A piece of information that is used to identify a computer or a user		
binary code	A sequence of 0 and 1 representing an algorithm that can be executed by a computer.	compiled code, machine code	
binary file	A file in a special format that can be executed by a computer. It is the result of a compilation	executable	
binary logic	the algebra of logical operators defined on booleans or binary systems	boolean algebra	
binary system	a counting system that has only two digits, 0 and 1. A fully defined algebra is defined on such system with logical and arithmetical operators. It is used by computers to perform calculations.		
binding	the association of a certain variable name to a certain memory location, virtual or real		
bit	BIrary digiT . The fundamental unit of measure of information. Can be 0 or 1		
body	A block of code that defines the behavior of a function		
boolean	A variable that can have only two values, true or false		
boot	the sequence of steps a computer (or any other system) does when starting		
byte	a set of 8 bits	word	

C.Glossary of terms

word	meaning	synonyms	related topics
call	the invocation of the name of a function or variable, or the use of any of these already defined, in a piece of code	invocation	declaration variable function
cluster	A set of computers interconnected, used for calculations.	computing cluster	
code	A structured ordered sequence of instructions, written in some programming language, that implement an algorithm.		
compilation	The process with which some code in a programming language becomes binary code.		
compiler	A software tool (app) to perform compilation		
CPU	Central Processing Unit, is the operational core of the computer. It's purpose is to execute instructions, at a rate defined by a CPU clock.		
credential	any information that can be used to identify a security principal.		see also authentication credential
data	information, stored in some format, that is either input or output of some algorithm.		
decimal system	the numeric system used by humans. It has 10 symbols to represent numbers		
declaration	the first time a variable or function name appears in some code and it is therefore added to the environment	definition	
digitization	the conversion of a continuous, analog information into a discrete one		
driver	a software whose purpose is to enable a computer to interact with some hardware		
dynamic binding	the binding of a variable name to a real memory location done by operating system libraries at runtime. Pointers affect the dynamic binding of a variable	dynamic relocation	binding pointer
editor	An app to modify files, mainly text files but not limited to	text editor	
environment	A table that contains names of variables and functions for each block of code or process and maps them to memory locations containing values or code.		

C.Glossary of terms

word	meaning	synonyms	related topics
executable	A compiled binary file that can be executed by a computer	binary file	
file	a sequence of 0 and 1 with a defined format that can be stored on a physical memory		
file browser	An app to browse files on a computer		
format	A set of rules that define how some data is structured		
frontend	A computer that serves as a point of access to a computing cluster.		
hardware	Any electronics that is related to a computer		
hash	A function that, given a certain set of input parameters S, returns a value $H(S)$ from a finite countable set of values, that is, a finite set of possible output values. It is used mainly to “summarize” the content of S.		
IDE	An Integrated Development Environment is a file editor with enhanced functionalities for code developers		
initialization	the assignment of a value to a variable	allocation, assignment	variable
internet browser	An app to browse the Internet		
interpreter	a software that compiles instructions of a language on the fly, without the need for the user to compile manually		
keyboard shortcut	A quick way to send a command to a computer just using the keyboard		
linker	A software tool (app) to perform linking		
linking	The process with which some code in a programming language is connected to existing libraries in an operating system		
memory location	a binary value that represents a location in memory	pointer	
Operating System	The collection of software that allows the computer to function, connects and coordinates the hardware, allows Human-Computer interaction		
OTP	One Time Password is an authentication credential that lasts for a very short time.		

C.Glossary of terms

word	meaning	synonyms	related topics
parameter	a piece of information passed to a command or a function to complete some task	option (for commands)	function command
password	A secret shared between a user and a computer system		
PKI	Public Key Infrastructure is a security paradigm in which an security principal is assigned two keys, a public one that can be disclosed to the whole world and a private one that should never be disclosed.		
pointer	a binary value that represent a location in memory	memory location	
program	The set of software components that define an application, it includes at least an executable file.	application, app	
programming language	A defined set of words and grammar rules that allow a developer to instruct a computer to do calculations, implementing some algorithm		
RAM	Random Access Memory is the volatile memory of a computer. It is completely wiped when a computer is turned off		
register	the hardware that contains a unit of memory in a computer		
ROM	Read-Only Memory. A chip that stores information that cannot be written directly by the user		
scope		visibility	
security principal	Anything that can be subject to a security protocol and uses authentication or authorization, for example a cluster user or a server	entity	
shell	An app that implements some kind of Command Line Interface (CLI)		
software	Any binary code that can be executed by a computer		
ssh	Secure SHell, a network communication protocol that features mutual authentication and secure encryption of transmitted data		
ssh hash	the hash of a ssh key		See also “hash” and “ssh key”
ssh key	A security credential used in the ssh protocol that is based on PKI. As such it consists of a public and a private key.		

C.Glossary of terms

word	meaning	synonyms	related topics
static binding	the binding of a variable name to a virtual memory location done by a compiler or an interpreter at compile time	static relocation	binding
terminal	A text or character base input output app or device that can be used to connect or interact with a computer using the command line		
text file	a special kind of file whose sequences of bytes is interpreted as characters by a computer		
type	the kind of information a variable can contain. It defines the amount of memory (size) the content of that variable will use	data type	variable definition
user	a person interacting with a computer system		
username	An identifier for a user in a computer system.		
variable	a name in the code associated to a data type and a memory location		
visibility	the presence of a variable in a given environment.		

D. References

Bibliography

aurora: Aurora Cluster at Lunarc, <https://www.lunarc.lu.se/resources/hardware/aurora/>
MATE: MATE desktop environment, <https://mate-desktop.org/>
tdgLFS: Linux filesystem , <https://www.2daygeek.com/wp-content/uploads/2016/10/linux-file-system-structure-final-4.png>

E.ChangeLog

Here I track the changes I am making to this document at each review.

I am following the GNU standard, see https://www.gnu.org/prep/standards/html_node/Style-of-Change-Logs.html#Style-of-Change-Logs

I suggest you follow such standard while writing the Final Project.

2022-09-03 Florido Paganelli <florido.paganelli@hep.lu.se>

Made some adjustments to ls examples in section 5.1 Command line syntax

2022-08-30 Florido Paganelli <florido.paganelli@hep.lu.se>

Exporting document as tagged PDF to improve accessibility
Fixed some pagination issues

2022-08-16 Florido Paganelli <florido.paganelli@hep.lu.se>

Updated info about how to obtain a Lunarc login.

2022-07-07 Florido Paganelli <florido.paganelli@hep.lu.se>

Reviewed all content for 2022.

2021-09-20 Florido Paganelli <florido.paganelli@hep.lu.se>

Added compilation concepts to manual glossary

2021-09-15 Florido Paganelli <florido.paganelli@hep.lu.se>

Added section A.6 SSH key pairs

2021-09-06 Florido Paganelli <florido.paganelli@hep.lu.se>

Added new words to the glossary

Changed "SSH key" to "SSH hash" in closing remarks of section 2.1

2021-09-05 Florido Paganelli <florido.paganelli@hep.lu.se>

Added a brief description of the authentication process at the end of section 2

2021-09-02 Florido Paganelli <florido.paganelli@hep.lu.se>

Changed sections from Advanced Topic on to have letters instead of numbers

2021-09-01 Florido Paganelli <florido.paganelli@hep.lu.se>

Added section on command line login tools

2021-08-31 Florido Paganelli <florido.paganelli@hep.lu.se>

Added section with CLI reference
Added section about WinSCP and FileZilla

2021-08-30 Florido Paganelli <florido.paganelli@hep.lu.se>

Updated info on ThinLinc file sharing

Added reference to the ChangeLog in the How to read this doc section
Added detailed information about the Caja Filebrowser
Added a section about the Linux filesystem
Added Glossary of terms

2021-08-29 Florido Paganelli <florido.paganelli@hep.lu.se>

Moved CodeBlocks info to Advanced Topics

E.ChangeLog

Moved section on connecting via textual terminal client to Advanced topics, still WIP

Added a section on how to open existing files with Pluma in alternative ways

Added an Advanced Topics section

* Added how to share remote files with ThinLinc

Added a ChangeLog section