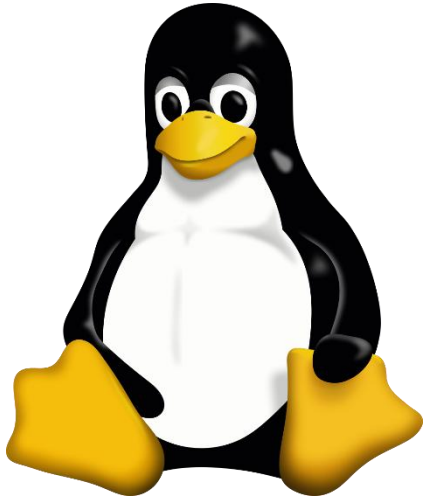
Introductory course on Linux command line interface

Kristijan Dekanić (SRCE)

Linux

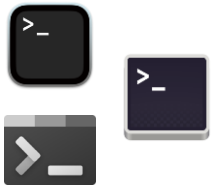


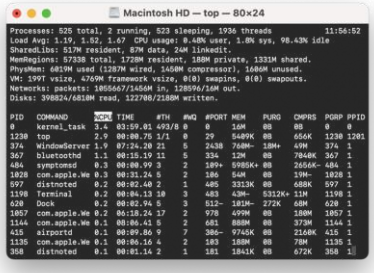
- **Linux** is the dominant operating system among the world's top 500 most powerful computers (supercomputers)
- When we refer to the "Linux operating system," we actually mean a **Linux distribution**, which consists of:
  - **The Linux kernel** (including device drivers)
  - GNU application software

Terminal & Shell

Terminal

- A desktop application where you can see the text interface used to interact with the operating system
- It allows users to enter text commands and receive text outputs in return

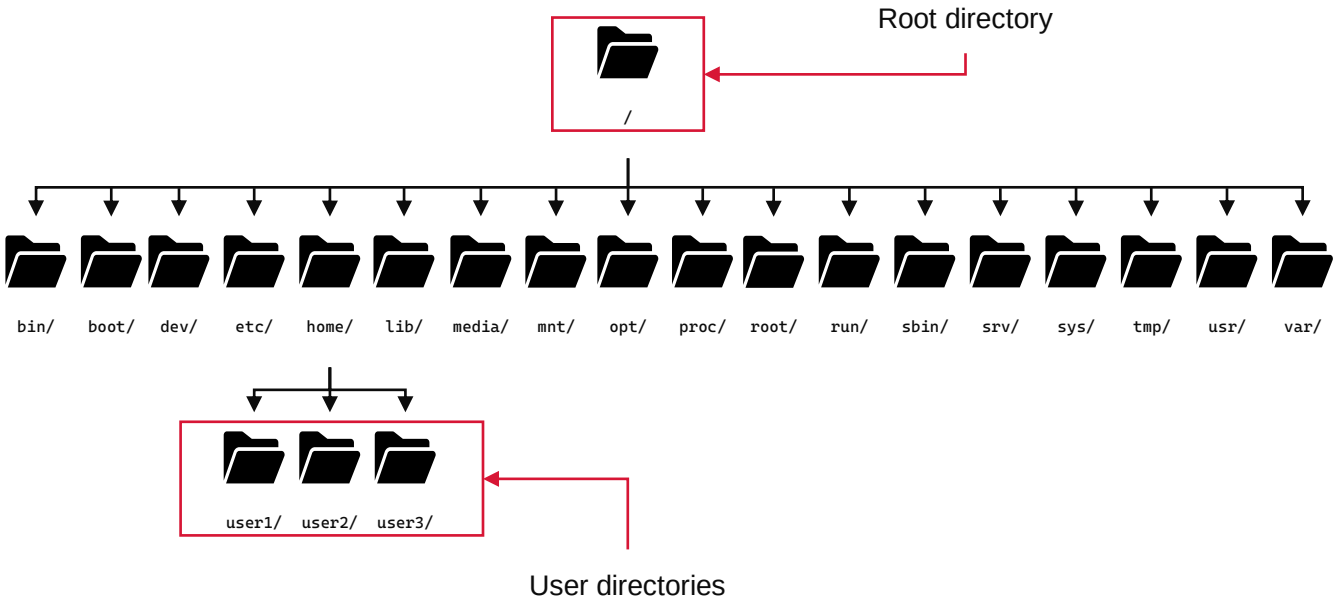




Shell

- A program that interprets and executes the commands you type into the terminal
- It interprets commands and communicates with the operating system to perform the requested actions

File system organization



SSH protocol

SSH protocol

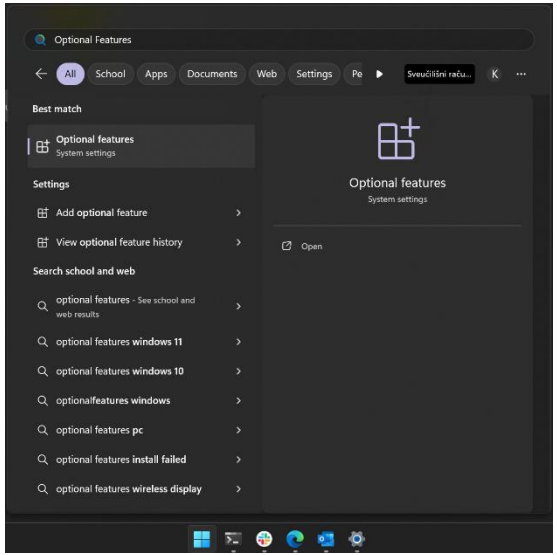
- Enables secure network communication between two computers
- It is used to connect to remote computers (servers) and securely execute commands on them

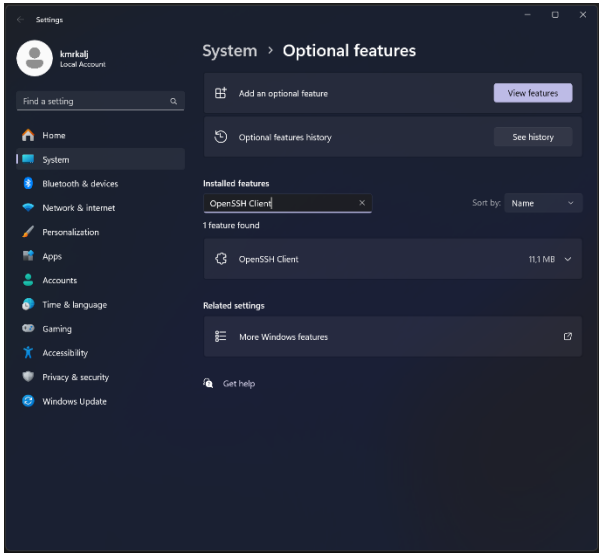
Private and public SSH keys

- A form of digital credentials used for passwordless authentication
- The private key is stored on the user's computer, while the public key is placed on the server
- When a user tries to establish an SSH connection, the key match is checked and the connection is established without the need to enter a password

Windows OpenSSH Client

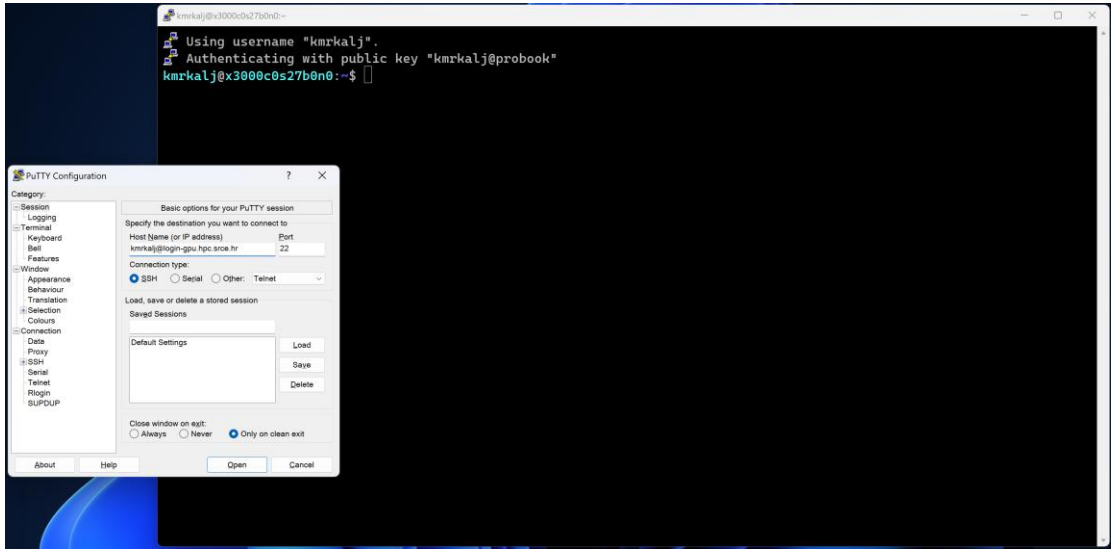
- To work with ssh commands in Windows Terminal, you need to install the OpenSSH Client feature





Windows PuTTY

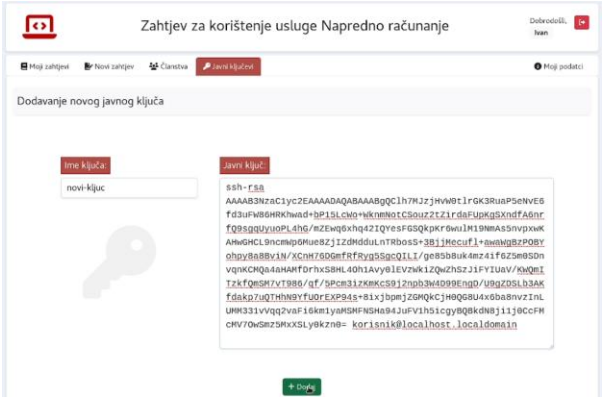
- A third-party free application used for remote communication via SSH protocol
- It is primarily used on Windows operating systems



Creating SSH Keys
Terminal

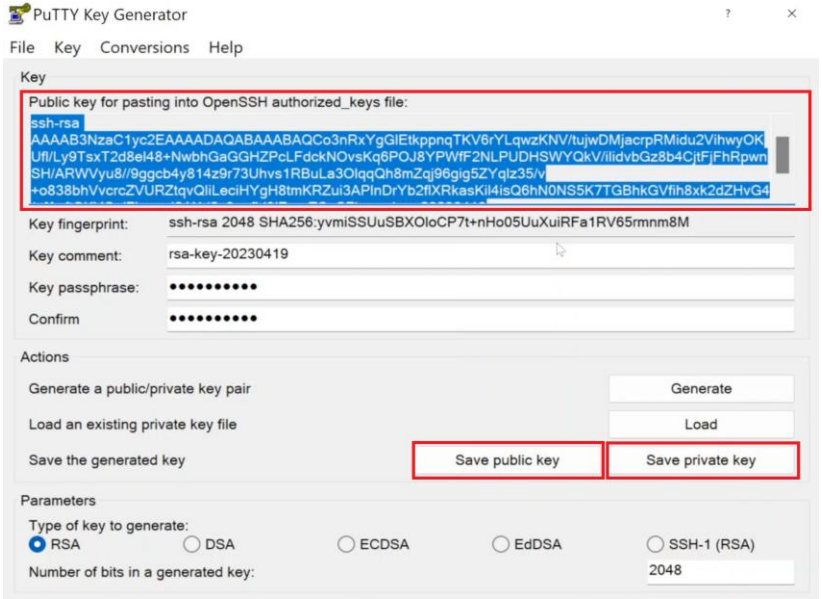
- To access Supek, you need to create a pair of SSH keys and add the public part of the key using the web application.
- **ssh-keygen** is a command to create a key pair
- The contents of the public part of the key (.pub file) need to be added using a web application

```
username@probook:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (~/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```



Creating SSH Keys
PuTTY Key Generator

- Comes with a full installation of the PuTTY app
- After clicking the "Generate" button in the application, it is necessary to save the private and public part of the key by pressing the "Save public key" and "Save private key" buttons
- The content in the box at the top of the window needs to be copied and uploaded using the web application



Access Nodes and Connection

Your Username

- Once you have successfully submitted your public key, you will receive an email containing your username and confirmation that you have set up your public key.

Access Nodes

- **Supek**
  - `login-cpu.hpc.srce.hr`
  - `login-gpu.hpc.srce.hr`

Connection Terminal

- The connection is done using the **ssh** command:

```
username@probook:~$ ssh username@login-cpu.hpc.srce.hr
```

- In the event of an error:

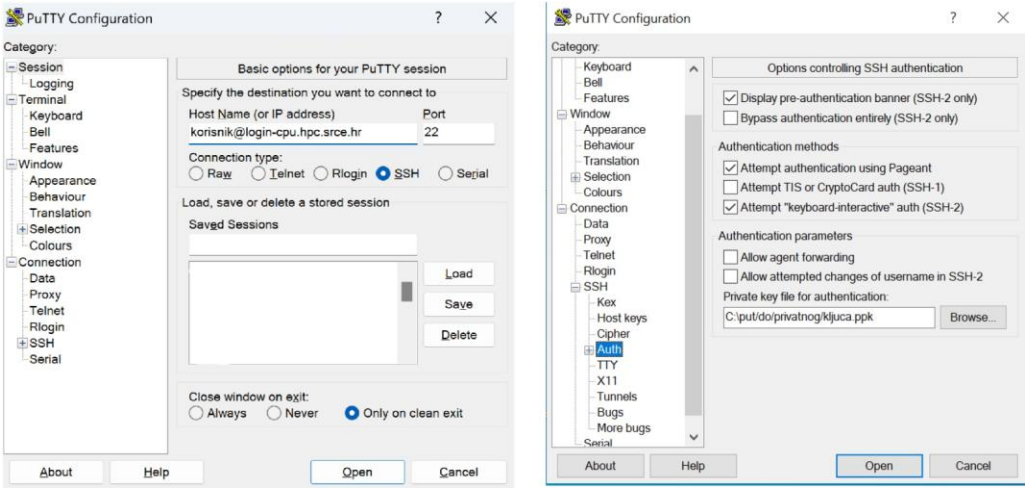
```
Permission denied (publickey,gssapi-keyex,gssapi-with-mic)
```

- Explicitly specify the path to the private key:

```
username@probook:~$ ssh -i ~/.ssh/id_rsa username@login-cpu.hpc.srce.hr
```

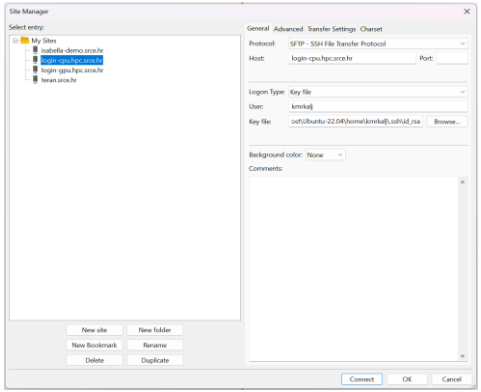
Connection
PuTTY

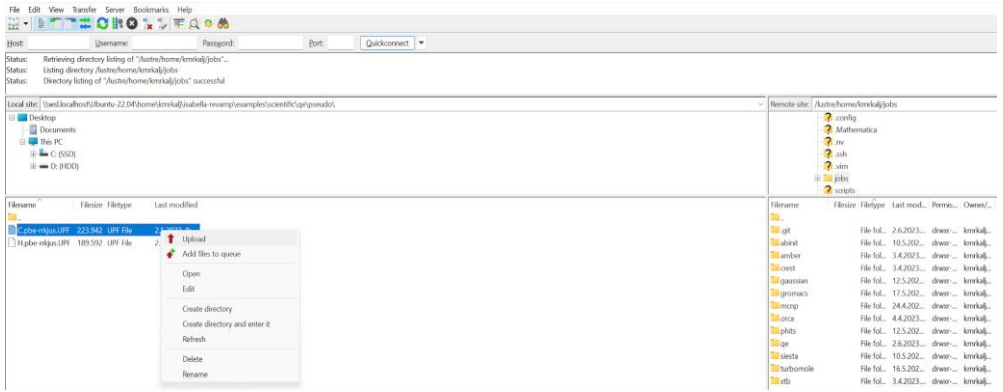
- The PuTTY tool can also be used to connect by typing the private key path



Copying files
GUI (FileZilla)

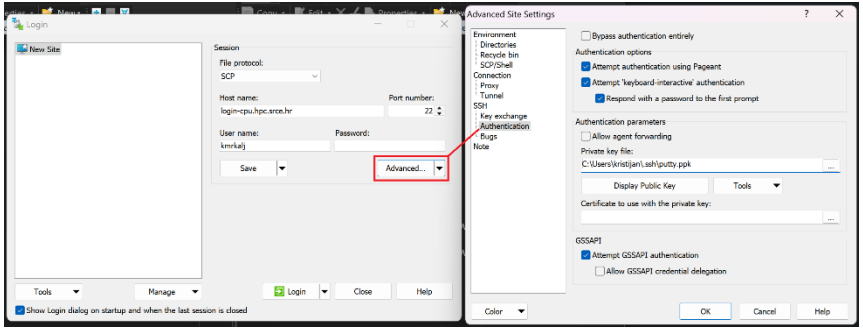
- All operating systems (Windows, Linux, macOS)
- A remote server (Supek) is first added via a menu from the toolbar, File → Site Manager
- File transfer is performed via the right-click drop-down menu, i.e. the Download and Upload options

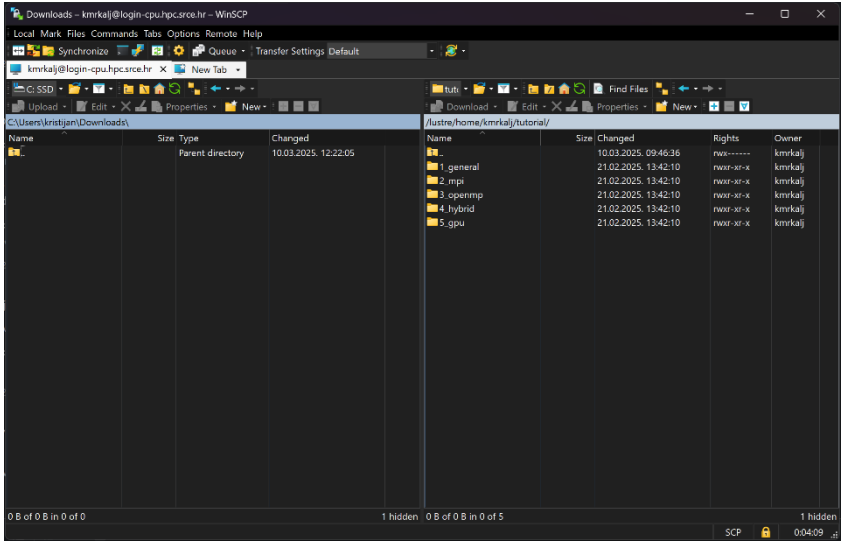




Copying files GUI (WinSCP)

- Windows
  - It's similar to FileZilla.





Useful keyboard shortcuts

Option	Explanation
`Tab`	Autocomplete command names or file names
`Ctrl c`	Interrupts the current process or command that is being executed
`Ctrl d`	Terminates the current shell session (equivalent to the `exit` command)
`Ctrl l`	Clean terminal screen

pwd

- Print the directory you're currently in

```
username@x3000c0s27b0n0:~$ pwd
/lustre/home/username
```

cd

- Moving to another directory
- Directories can be defined as:
  - absolutely, from the original directory
  - relative to the current directory.

```
username@x3000c0s27b0n0:~$ cd /opt/nvidia/

username@x3000c0s27b0n0:/opt/nvidia$ cd ~

username@x3000c0s27b0n0:~$ cd linux

username@x3000c0s27b0n0:~/ccd2026/linux$
```

Special character	Explanation
~	User directory, /home/user/<…>
.	Current directory
..	The parent directory

ls

- Print the contents of the directory

Option	Argument	Explanation
-a	-	Includes both hidden files and directories
-l	-	A longer, more extensive record

```
username@x3000c0s27b0n0:~/ccd2026/linux$ ls -l
total 4
-rw-r--r-- 1 username hpc 11 Mar  6 14:59 copyme.txt
drwxr-xr-x 2 username hpc  0 Mar  6 14:48 deleteme
-rw-r--r-- 1 username hpc 11 Mar  6 14:59 deleteme.txt
-rw-r--r-- 1 username hpc 30 Mar  6 14:57 file_1.txt
-rw-r--r-- 1 username hpc 30 Mar  6 14:57 file_2.txt
-rw-r--r-- 1 username hpc 30 Mar  6 14:57 file_3.txt
-rw-r--r-- 1 username hpc 13 Mar  6 14:59 moveme.txt
-rw-r--r-- 1 username hpc 13 Mar  6 15:03 renameme.txt

username@x3000c0s27b0n0:~/ccd2026/linux$ ls -l /opt/nvidia/
total 0
drwxr-xr-x 4 root root 45 Oct 29  2022 hpc_sdk
drwxr-xr-x 3 root root 22 Mar 25 14:33 nsight-compute
drwxr-xr-x 3 root root 22 Mar 25 14:33 nsight-systems
```

cat

- Print the contents of one or more (text) files

```
username@x3000c0s27b0n0:~/ccd2026/linux$ cat file_1.txt file_2.txt
This is the content of the file no. 1!
This is the content of the file no. 2!
```

touch

- Creates an empty text file, if it does not exist
- Refreshes the date the file was last accessed and edited, if it existed

```
username@x3000c0s27b0n0:~/ccd2026/linux$ touch touchme.txt
```

mkdir

- Creating one or more empty directories

Option	Argument	Explanation
-p	-	It also creates "parent" directories

```
username@x3000c0s27b0n0:~/ccd2026/linux$ mkdir dir_1

username@x3000c0s27b0n0:~/ccd2026/linux$ mkdir -p dir_2/dir_21
```

rm

- Delete one or more files

Option	Argument	Explanation
`-r`	–	Recursively deletes directories, i.e. deletes directories and their contents
`-v`	–	Verbose deletion, i.e. deletion with the printing of deleted content
`-i`	–	Interactive deletion, i.e. deletion with confirmation of action

```
username@x3000c0s27b0n0:~/ccd2026/linux$ rm -v deleteme.txt
removed 'deleteme.txt'

username@x3000c0s27b0n0:~/ccd2026/linux$ rm -rv deleteme
removed 'deleteme/trash_2.tmp'
removed 'deleteme/trash_1.tmp'
removed 'deleteme/trash_3.tmp'
removed directory 'deleteme'
```

cp

```
username@x3000c0s27b0n0:~/ccd2026/linux$ cp -v copyme.txt copy.txt
'copyme.txt' -> 'copy.txt'

username@x3000c0s27b0n0:~/ccd2026/linux$ cp -v copyme.txt copy.txt dir_1/
'copyme.txt' -> 'dir_1/copyme.txt'
'copy.txt' -> 'dir_1/copy.txt'
```

- Copy a file
- Copy one or more files to the final directory

Option	Argument	Explanation
`-r`	–	Recursively copies directories, that is, copies directories and their contents
`-v`	–	Verbose copying, i.e. copying with printing of copied content
`-i`	–	Interactive copying, i.e. copying with action confirmation

mv

- Rename a file
- Move one or more files to the final directory

```
username@x3000c0s27b0n0:~/ccd2026/linux$ mv -v renameme.txt renamed.txt
renamed 'renameme.txt' -> 'renamed.txt'

username@x3000c0s27b0n0:~/ccd2026/linux$ mv -v moveme.txt dir_2/
renamed 'moveme.txt' -> 'dir_2/moveme.txt'
```

Option	Argument	Explanation
`-v`	–	Verbose renaming/moving, i.e. with printing changes to content
`-i`	–	Interactive renaming/moving, i.e. with action confirmation

Basic terminology

Standard Output

- That part of the output of a command-line command that refers to everything that is not an error or a warning

Standard Error

- That part of the output of a command-line command that refers to everything that is an error or a warning

Redirect

- Standard output and standard error can be redirected from the "screen" to separate files
- A double parenthesis, >>, allows you to append a file, while a single parenthesis, >, will overwrite its contents

Character	Argument	Explanation
1>	<file>	Redirecting std.output
2>	<file>	Redirecting std.error
&>	<file>	Redirecting std.output & std.error

```
username@x3000c0s27b0n0:~/ccd2026/linux$ cat file_1.txt deleteme.txt
This is the content of the file no. 1!
cat: deleteme.txt: No such file or directory

username@x3000c0s27b0n0:~/ccd2026/linux$ cat file_1.txt deleteme.txt 1>out.txt 2>err.txt

username@x3000c0s27b0n0:~/ccd2026/linux$ cat out.txt
This is the content of the file no. 1!

username@x3000c0s27b0n0:~/ccd2026/linux$ cat err.txt
cat: deleteme.txt: No such file or directory

username@x3000c0s27b0n0:~/ccd2026/linux$ cat file_1.txt deleteme.txt &>out_err.txt

username@x3000c0s27b0n0:~/ccd2026/linux$ cat out_err.txt
This is the content of the file no. 1!
cat: deleteme.txt: No such file or directory
```

Basic terminology

Local variables

- Variables that are defined within the current shell, i.e. shell session and are only available within that shell
- They will not be available outside of the current shell, i.e. in other shells that are created from it or in scripts that run from it ("daughter shells")
- An example of a local variable is a variable that you set directly while working in a shell

Environment variables

- Variables that are defined within the current shell, i.e. shell session, but which are also available in other shells created from it ("daughter shells")
- Examples of environment variables in a shell include:
  - a variable that you set while working in the shell, but which you then export
  - environment variables that are set when logging in to the system, and that affect applications available from the shell

Common environment variables

- **PATH** is a variable that contains a list of directories (separated by a colon) in which the operating system searches for executable files (commands)
  - If you want to run a command, the operating system will look for it in each of the directories
- **HOME** is a variable that contains the path to the user's directory
- **USER** is a variable that contains the user name of the currently logged in user
- **HOSTNAME** is a variable that contains the name of the computer (host) to which you are currently logged on

export

- Sets and exports a variable to the environment
- Exports the local variable to the environment

```
username@x3000c0s27b0n0:~/ccd2026/linux$ VAR=5

username@x3000c0s27b0n0:~/ccd2026/linux$ export VAR
```

ili

```
username@x3000c0s27b0n0:~/ccd2026/linux$ export VAR=5
```

Print a variable value

`echo $VARIABLE`

- Print the value of a specific variable

```
username@x3000c0s27b0n0:~/ccd2026/linux$ echo $VAR
5
```

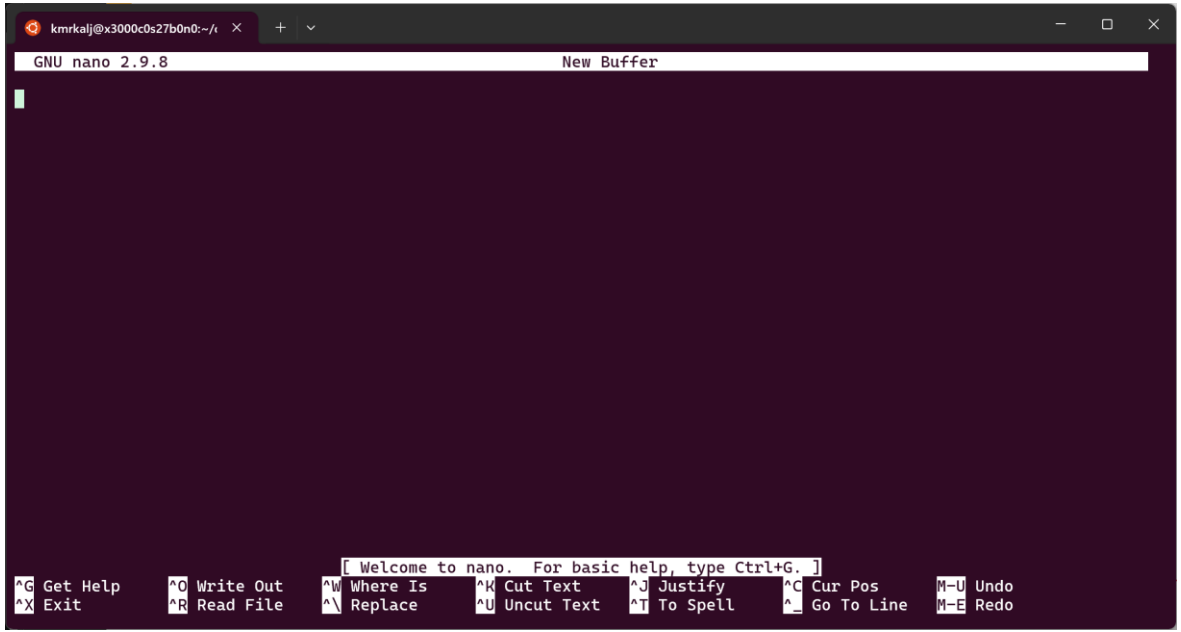
`printenv`

- Print all environment variables

```
username@x3000c0s27b0n0:~/ccd2026/linux$ printenv
LOGNAME=username
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/2005/bus
XDG_RUNTIME_DIR=/run/user/2005
MODULEPATH_modshare=/usr/share/modulefiles:1:/usr/share/Modules/modu
lefiles:1:/etc/modulefiles:1
MODULEPATH_ROOT=/usr/share/modulefiles
PATH=/lustre/home/username/.local/bin:/lustre/home/username/bin:/opt
/clmgr/sbin:/opt/clmgr/bin:/opt/sgi/sbin:/opt/sgi/bin:/usr/share/Mod
ules/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/c3/b
in:/opt/pbs/bin:/sbin:/bin:/lustre/home/username/.local/go/bin:/lust
re/home/username/.fzf/bin
PS1=\[\e[01;36m\]\u\[\e[m\]@\[\e[01;36m\]\h\[\e[m\]:\[\e[01;34m\]\w\
[\e[m\]\[\e[01;33m\]`parse_git_branch`\[\e[m\]$
MODULESHOME=/usr/share/lmod/lmod
LMOD_SETTARG_FULL_SUPPORT=no
HISTSIZE=1000
```

nano

- Simple
- Small number of options
- User-friendly



Shell Script

- A text file containing a series of shell commands that are executed one after the other
- They make it easy to automate tasks on Unix-like operating systems
- These can be tasks such as processing text files, managing files and directories, executing programs with specific arguments, etc.

```
#!/bin/bash

# Print a message to the user
echo "This is my first shell script."

# Display the current date and time
echo "Current date and time:"
date
```

Creating and running a shell script

1. Write a script using some text editor, such as Vim, Nano, or any other.
2. When you're done writing the script, save it to a file, such as **my_script.sh**

4. Run the script by typing by calling the interpreter

```
username@x3000c0s27b0n0:~/ccd2026/linux$ bash moja_skripta.sh
```

5. Once you run the script, the shell will execute the commands that are defined in the script one by one, following the order you defined