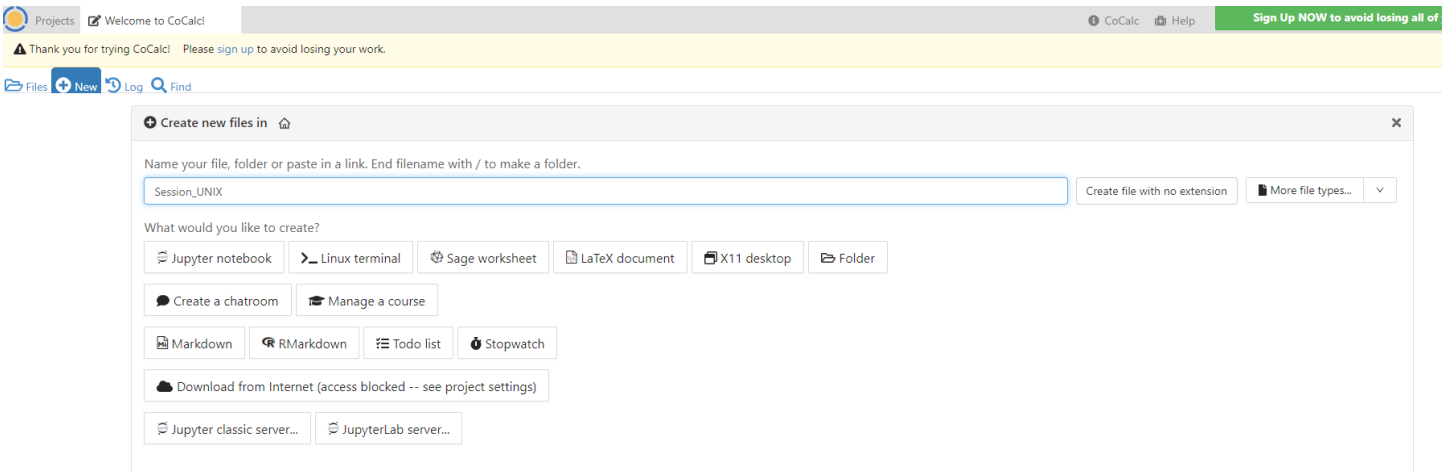
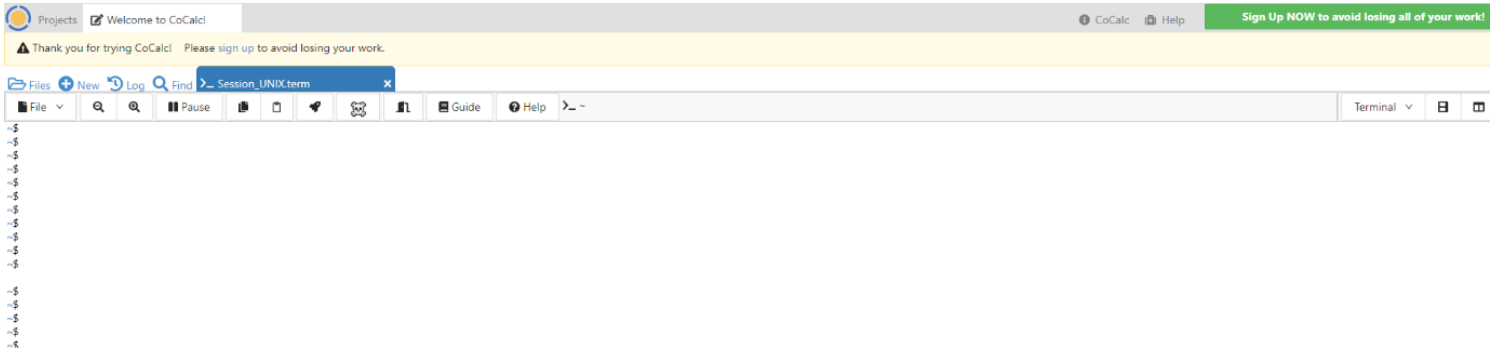
Workshop on UNIX command line

Note: if you do not have access to a UNIX machine after the workshop, you may use an online emulator, available at https://cocalc.com/ . You can create a new session simply by creating a new Linux terminal (see screen capture below):



Select the Linux Terminal. You should end up on a page looking like this:



This window is the terminal: this is where we will launch our commands.

Part I: Creating files and navigating in folders using the command line

The most difficult part in using the command line is that we cannot use the mouse, nor can we rely on a user-friendly presentation of our Desktop, folders and files.

Let's start by looking at where we are. Type the following command:

```
pwd
```

What do you see? Can you guess what it means?

In UNIX, most functions have a man page, which describes what the function does, and what its options are. Type this command and see what happens:

```
man pwd
```

Let's try now to move and create new folders. Type the following commands:

```
mkdir new_directory
mkdir new_directory_2
mkdir new_directory_3
ls
rm -r new_directory_3
ls
```

mkdir stands for "make directory". It creates a new folder. The *ls* command stands for "list": it lists the content of your current folder. At last rm stands for "remove": it deletes the directory. The –r option stands for recursive: it destroys the directory and its content (you can check what happens if you do not add the –r option).

To navigate between different folders, we need to change the directory. This is done with the **cd** command (for Change Directory). Try this:

```
cd new_directory
pwd
cd ..
pwd
cd new_directory/
pwd
cd ../new_directory_2
pwd
```

See what happens?

Task: Create a folder named subdirectory inside new_directory, enter it, and check that you are in the right place using the commands described earlier. Then go back to the folder containing new_directory and new_directory_2.

Part II: Basic text editing with nano, other commands.

So far, we simply created folders, but did not put anything inside. However, you can create text files using the nano command. Type this:

```
nano new_file.fa
```

Then copy and paste the following text:

```
>Protein
MAQQWSLQRLAGRHPQDSYEDSTQSSIFTYTNSNSTRG
PFEGPNYHIAPRWVYHLTSVWMIFVVTASVFTNGLVLA
ATMKFKKLRHPLNWILVNLAVADLAETVIASTISIVN
QVSGYFVLGHPMCVLEGYTVSLCGITGLWSLAIISWE
RWMVVCKPFGNVRFDAKLAIVGIAFSWIWAAVWTAPP
IFGWSRYWPHGLKTSCGPDVFSGSSYPGVQSYMIVLM
VTCCIIPLAIIMLCYLQVWLAIRAVAKQQKESESTQK
AEKEVTRMVVVMIFAYCVCWGPYTFFACFAAANPGYA
FHPLMAALPAYFAKSATIYNPVIYVFMNRQFRNCILQ
LFGKKVDDGSELSSASKTEVSSVSSVSPA
```

List of commands in nano

Command	***Function***
ctrl+o	save file
ctrl+x	close file
alt+/	go to end of the file
ctrl+a	go to start of the line
ctrl+e	go to end of the line
ctrl+c	show line number
ctrl+_	go to line number
ctrl+w	find matching word
alt+w	find next match
ctrl+\	find and replace

Use these commands to save and close the file. You can navigate through the file using the arrows on your keyboard.

Another text editor, much less intuitive (it is possible), is called vim. If you have time, I invite you to consult this link for more information:

http://yannesposito.com/Scratch/en/blog/Learn-Vim-Progressively/

You can also use the echo command to output a string of characters. For example, try the following commands.

```
echo "ATGWTCWC"
echo "ATGWTCWC" > DNA_string.txt
ls
echo "CTWCCGGC"
echo "CTWCCGGC" >> DNA_string.txt
ls –s DNA_string.txt
```

You should see two files and two folders in the current directory. You can use different commands to examine your file without editing it. These commands include **less** (to inspect your file), **head** and **tail**. ***ls –s*** gives you the size of your file (or of the files in a folder).

Task: Use the man pages for head and tail to find how to output the first and the last row of the new_file.fa file.

Task: Use less to inspect the file DNA_string.txt. What do you notice? Try to rerun the *echo* commands but replace `>>` by `>`. Inspect the file again. What can you say about the meaning of '> 'and '>>' ?

Part III: Extracting rows and columns, listing, merging, renaming, and moving files

You can also extract specific rows in a file by using the *sed* command

```
sed -n 2p new_file.fa
sed -n 2,4p new_file.fa
sed -n -e 1,2p -e 4p new_file.fa
```

Note that the *sed* command can also be used to replace a string of characters by another one. For example, to replace all W by A in the DNA_string.txt file, you can use the following command:

```
sed "s/W/A/g" DNA_string.txt
```

Try the same command but remove the "g". What happens?

Task: Try to replace all T characters by W in the file DNA_string.txt. Send the output to a new file called edited_DNA_string.txt using the ">" character. Inspect the file with *less*. Then, use the *rm* command to delete the original file.

Use nano to create two new files Treatment1.txt and Treatment2.txt containing the following text (tab-delimited):

Treatment	**Statistics**
Treatment_1	**0.5**
Treatment_2	**0.1**
Treatment_3	**0.5**
Treatment_4	**0.8**
Treatment_5	**0.9**

And

Treatment	**Statistics**
Treatment_6	**0.1**
Treatment_7	**0.3**
Treatment_8	**0.5**
Treatment_9	**1.8**
Treatment_10	**1.2**

Try the following commands:

```
cut -f2 Treatment1.txt
cat Treatment1.txt
cat Treatment1.txt Treatment2.txt
tail -n5 Treatment2.txt | cat Treatment1.txt -
```

The "|" character is the pipe: it redirects the output of a command as an input for the command that is on the right. Note the "-" character in the last cat command above: it means that Treatment1.txt must be con***cat***enated with the output of *tail*.

Note that we can paste files by column. For example:

```
paste Treatment1.txt Treatment2.txt
```

Task: How would you create a file with three columns that contain Treatment1.txt and the Statistics field of Treatment2.txt ?

At last, let's have a look at how we can move and rename files. The command is mv, and it can also be used to rename a file. For example:

```
mv *.txt new_directory
ls new_directory
cd new_directory
mv Treatment1.txt New_name.txt
ls
cd ..
```

Note the difference between the first mv command, which moves files to a folder, and the second mv command that changes the name of the file. Note also that we used the * character, which stands for any string of character. Here, we moved all files ending with ".txt".

Part IV: Introduction to grep.

A very useful tool in UNIX is called grep, which stands for Global regular expression print. It looks for patterns in a file. For example try the following commands:

```
grep MAQ new_file.fa
grep -c MAQ new_file.fa
grep -n MAQ new_file.fa
grep -A2 MAQ new_file.fa
grep -B1 MAQ new_file.fa
```

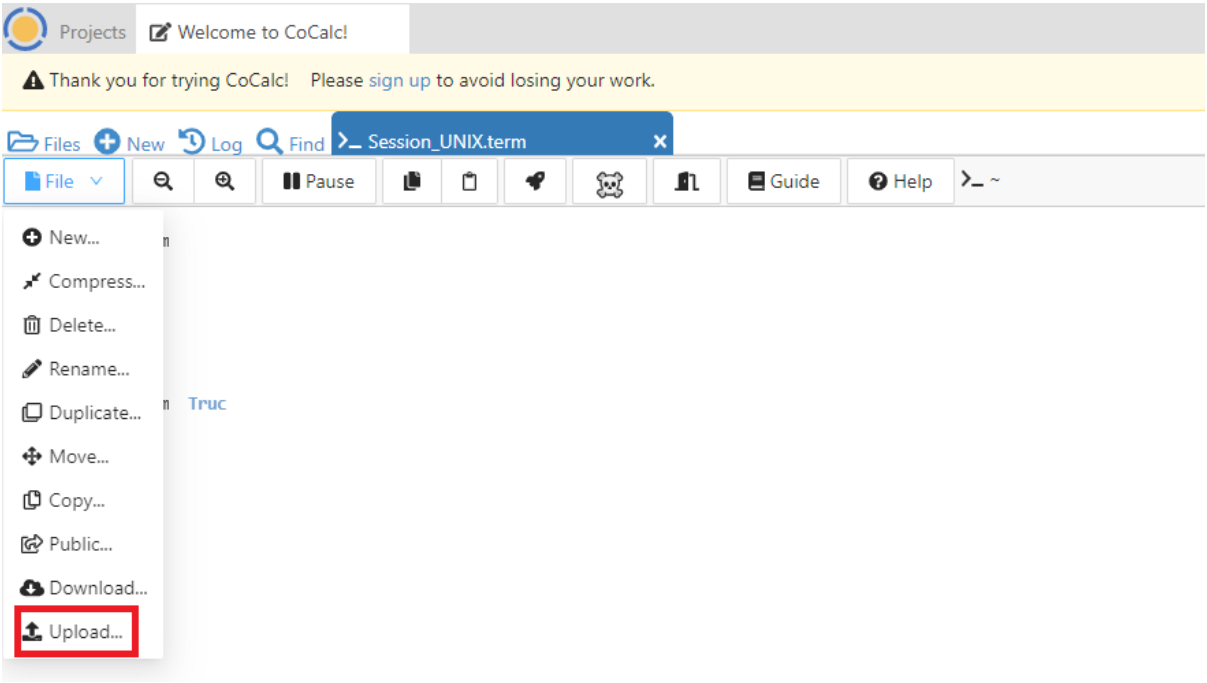
Try to make sense of the different options.
You can also look for patterns that are stored in a file. **Create a file named patterns.txt containing the following rows:**

VADLAET
FAKS

And type the following command:

```
grep -f patterns.txt new_file.fa
grep -f patterns.txt -c MAQ new_file.fa
grep -f patterns.txt -n MAQ new_file.fa
```

For the last task, we will upload a large file in the environment. The file can be found on Moodle (sequence.fasta). You can upload the file by selecting file and upload.



Task: Move this file into the folder new_directory_2. List it and check its size. Then, use *grep* to count how many different sequences are in the file (remember that sequences names in fasta files start with a ">" character).

Cheat sheet

You can find a list of UNIX commands at this website:

https://bioinformaticsworkbook.org/Appendix/Unix/UnixCheatSheet.html#gsc.tab=0