# TP n°1 Unix [solution]

samuel.ortion@universite-paris-saclay.fr

2026-09

## Introduction

### To practice on the Unix shell

You should have access to the Ubuntu session on the computer in the room, with your student number and student password. If you have a Unix-like OS (e.g., MacOS, GNU/Linux) on your laptop, you can also use it if you prefer. If you have Windows installed on your machine, you still have multiple options if you want to work on your machine. I listed some of these options on `https://s.ortion.fr/w/course/mssb-computer-refresher/introduction.html`

### To take some notes

You should keep track of the commands you used for this practical exercise. You should avoid using a text processing application, such as LibreOffice or Word, because it might introduce unwanted character in the commands. Prefer using a plain text file, or, even better: use a standard plain text format for your notes such as a Markdown file or org-mode for instance.

## Exercise 1. A first step in the Unix world

1. Check your unix username in two different ways
   (a) Use the command `whoami`
   (b) Check the content of the environment variable `$USER`

   **solution**:

```
$ whoami
sortion
$ echo $USER
sortion
```

2. Create a folder with `mkdir` in your home directory for this practical exercise.

   **solution**:

   ```
   mkdir unix_2026
   mkdir unix_2026/tp1
   # or, with the option -p of mkdir,
   # we can create both directory in one command
   mkdir -p unix_2026/tp1
   ```

3. Move to this newly created directory with the `cd` command

   **solution**:

   ```
   cd unix_2026/tp1
   ```

4. Check your current working directory with `pwd`

   **solution**:

   ```
   pwd
   ```

5. Your Unix machine probably have the `curl` or `wget` command line included. This subject document should be available online at `https://s.ortion.fr/w/course/mssb-computer-refresher/unix/tp/tp1/tp1.org` and `https://s.ortion.fr/w/course/mssb-computer-refresher/unix/tp/tp1/tp1.pdf` if you prefer to get a PDF file. Using `wget` or `curl`, download this document in your local folder.

   **solution**:

   ```
   # with wget
   wget https://s.ortion.fr/w/course/mssb-computer-refresher/unix/tp/tp1/tp1.org
   # with curl
   curl -O tp1.org https://s.ortion.fr/w/course/mssb-computer-refresher/unix/tp/tp1
   ```

6. Using `ls`, check that this is correctly downloaded and present in your current directory.

   **solution**:

```
ls
```

7. Open the file you downloaded from the terminal. If you chose to download the PDF rather than the Org-mode plain text source, you will need to use a PDF viewer.

   **solution**:

   ```
   # with less
   less tp1.org
   # with more
   more tp1.org
   ```

8. Using `grep` and a regular expression count how many questions there are in the `tp1.org` file. Each question has a number. The line of a question hence starts with the question number followed by a dot.

   **solution**:

   ```
   grep -cE "^[0-9]+\." ./tp1.org
   ```

   Let's dissect the regular expression. We start with '^' to match only the lines that starts with this regular expression, otherwise, every lines that would contain a number followed by a dot (for instance, a real number), would be counted as a question line. Then, to match a number with use a range of character `[0-9]` (any digit from 0 to 9). We add a + sign to match also any questions that has a number with more than one digit. Then, we the dot; but because the dot also represents a class of character by default in regular expressions, we must escape the '.' character with the backslash character : "\.".

## Exercise 2. A grep on grape

Not a long time ago took place the grape harvest. France is one of the main grape and wine producers worldwide.

The FranceAgriMer institution is a national organisation responsible of data analysis on French agricultural activities, and to apply the European common agricultural policy in France. We will use a reference document listing the grape varieties that we can find in the market. Unfortunately the reference table is stored in a inconvenient PDF format. To make your life easier, I extracted the first 100 lines of this table into a plain text file

in the comma-separated values (CSV) format and translated the comment and table header: `./data/agrimer_grape.csv`

source

1. Remember that with `grep` you can search for a pattern in a text file with `grep PATTERN FILE`. For instance, I could write the following command to find all occurrences of the term `Chasselas` (a well-known grape variety) in the file:

   ```
   grep Chasselas ./data/agrimer_grape.csv
   ```

   I can also count how many lines have this pattern term with the grep option `-c`:

   ```
   grep -c Chasselas ./data/agrimer_grape.csv
   ```

   ```
   2
   ```

   I kept the PDF file header in the csv in a "comment" section, starting with the character '#'. With grep you can also show all lines that don't match the pattern (we sort of invert the grep query, with `-v`).

   How would you filter out the comment lines from the csv file with a grep command ?

   **solution**:

   ```
   grep -v -E "^#" ./data/agrimer_grape.csv
   ```

   We can redirect the output of the `grep` command to a file with the superior sign ">":

   ```
   grep -v -E "^#" ./data/agrimer_grape.csv > /tmp/uncommented_agrimer_grape.csv
   ```

2. If you used only "#" as a grep pattern, think of what would occur if a record line contained a hashtag.

   With the grep option `-E` you can specify a more advanced regular expression pattern (`-E` means extended regular expression support). What regular expression would match only the lines that starts with the hashtag '#' character ?

   Use this to fix your previous command.

3. Count how many grape varieties are maintained by an institution in France (FR).

   **solution**:

   ```
   cut -d, -f4 ./data/agrimer_grape.csv | grep -c "(FR)"
   ```

   ```
   86
   ```

4. Check whether there is a grape variety that is not maintained in France.

   **solution**:

   ```
   cut -d, -f4 ./data/agrimer_grape.csv | grep -vc "(FR)"
   ```

   ```
   12
   ```

5. In fact, while manually editing the file, I messed up with some lines… Those lines does not have the expected number of columns (4). Can you find these lines?

   **solution**:

   ```
   awk -F, 'NF != 4' /tmp/uncommented_agrimer_grape.csv
   ```

## Exercise 3. An executable bash script

1. Open your favorite text editor (not Word, but a plain text editor), and create a file with the extension `.sh`.
2. Add a shebang in the header of this script to use the `bash` shell interpreter.
3. The file `/proc/cpuinfo` contains information on the CPU of your machine. Add a command to extract the `model name` of your CPU from this file.
4. Add a command to count how many available CPU of this kind there are on your system.
5. Make your script executable for your user and group.
6. Execute your script

7. Add the folder that contains your script to the PATH variable.

8. Check that you can execute your script as a command, from any folder.

**solution**:

I am sorry for Mac users: it seems that this file does not exist on MacOS machines.

An example script can be found on `bin/script.sh`.

Let us have a look at the commands I used here.

First, a remark on the syntax `$(command)`. It can be used to put the result of a command inside a bash variable, as I did here for `model_name` and `number_of_cpu`.

Then, let's have a look at the pipeline I used to extract the CPU model name:

```
model_name=$(grep "model name" /proc/cpuinfo | head -1 | cut -d: -f 2 )
```

I use `grep` to extract the lines that contains the "model name" string. Then, I keep only one of them with `head -1`. Finally, with `cut`, I extract the model name: the second column, where the columns are separated by ":".

Next, for the number of CPU, one possibility is to count how many times the model name appears in the file.

So, as usual, we can use grep:

```
number_of_cpu=$(grep -c "${model_name}" /proc/cpuinfo)
```

I use the content of the `model_name` variable as the pattern searched by `grep`.

Another possibility is to use `awk`: For instance, we could write this `awk` command.

```
awk -F':' '
    /^model name/ {
        model_name=$2;
        count_cpu[model_name]++
    }
    END {
        for (cpu_model in count_cpu) {
            printf "this machine has %d CPUs of model %s",
                count_cpu[cpu_model],
```

```
                cpu_model

    }
}' /proc/cpuinfo
```

Moreover, this approach have the advantage to also work in the case there were multiple different CPU models on the machine too.

To make the script executable, run:

```
chmod +x ./bin/script.sh
```

We can now execute it as:

```
./bin/script.sh
```

We can add the bin folder to the PATH variable:

```
export PATH=/path/to/my/bin/folder:$PATH
```

and we would be able to run the script with:

```
script.sh
```

## Exercise 4. Getting help

A lot of information is available on the web on Linux and other Unix-like systems. Most of command line tools comes with a `--help` option to show a description of the options available, e.g.

```
grep --help
```

Most of standard Unix commands also comes with an entry in the Manual. To access the manual from the terminal, is often as easy as typing `man <the name of the command>`, e.g.,

```
man grep
```

1. Check the manual entry of `cp`. What `cp` option can we use to recursively copy the content of a directory? To quit the manual, type `q`.

**solution**: We can use `cp -r` to recursively copy the content of a folder.

## Exercise 5. Internet and SSH

1. Can you find the ip address of your current machine from the terminal?
2. Knowing your Unix username and the address of your machine, and assuming that a SSH server is running on this machine, what would be the SSH command you would use to open a shell from a remote client to this machine?

**solution**

The solution to the first question really depends on your Unix system.

For modern Linux machine the IP address of the machine can often be obtained with:

```
ip a
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qle
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
3: eno1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group
    link/ether ac:f4:66:51:6d:f9 brd ff:ff:ff:ff:ff:ff
    altname enp0s31f6
    altname enxacf466516df9
4: wlp0s20f3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
    link/ether 02:a5:de:b7:58:48 brd ff:ff:ff:ff:ff:ff permaddr 80:e4:ba:8b:67:17
    altname wlx80e4ba8b6717
    inet 172.24.215.209/20 brd 172.24.223.255 scope global dynamic noprefixroute wlp0
       valid_lft 256315sec preferred_lft 256315sec
    inet6 fe80::d9b6:2e56:c14c:cc96/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
10: enp0s13f0u1u4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
    link/ether ac:f4:66:5e:86:e5 brd ff:ff:ff:ff:ff:ff
    altname enxacf4665e86e5
    inet 172.23.145.80/20 brd 172.23.159.255 scope global dynamic noprefixroute enp0s
       valid_lft 256318sec preferred_lft 256318sec
    inet6 fe80::be64:2299:bbff:57a9/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Here we can get IPv4 address for each of our connection device. For instance, in this case, my ethernet device `enp0s13f0u1u4` has the ipv4 address 172.23.145.80.

So, if I had an OpenSSH running on my machine, and my Unix username on this machine was `samuel`, I could connect with `ssh` with the following command:

```
ssh samuel@127.23.145.80
```