



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

First steps with the terminal/shell

Integration week – TAF DCL

Note: This document has been translated using automated tools. The original version was written in French, and the English version has not (yet) been thoroughly verified.

Objective

The goal of this (mini) lab session is to discover the terminal and the shell.

The command line interface (CLI)

Working with graphical user interfaces is often easy and intuitive, even if they are ultimately quite changeable (the number of buttons to click, their names, and their functions can vary from one version to another). However, in Unix, there are a large number of tools that are preferably accessible via the command line in a terminal or terminal emulator. These are the Unix commands. However, in Unix there are a large number of tools that are best accessed via the *command line* in a terminal or terminal emulator. These are the *Unix commands*. It is often preferable to use them for fast and efficient work, but learning how to use them requires some effort.

1 Shell and command

Open a terminal emulator: in the top menu bar, select **Applications**. There you will find the **Terminal** application either in the **Favorites** tab or in the **Utilities**¹

A window appears on the screen. It displays a small, obscure text called the *prompt* which, as its name suggests, prompts you to enter a command. In fact, you have two things at this point: you have (1) a graphical application that can display a window, manage a small scroll bar, display ASCII characters and under different encodings, and you have (2) a command interpreter called Shell.

¹This may vary from one version of the graphical environment to another. Sometimes it is **Accessories** or **System**.

The Shell associated with your window asks you to enter a command using the keyboard. When you do so, it will analyze this command and then schedule its execution. This command can be complex; it usually corresponds to an executable file, but its execution can be configured in different ways. The analysis prior to execution consists of taking these settings into account.

2 The Unix commands and the Shell

2.1 . and .. directories

In the terminal emulation window, type the command `ls -a1`. You should see a list of the contents of the directory you are in, including hidden files and directories. Compared to the list displayed in the Nautilus browser, you will also see the directories `.` and `..`.

Execute the following commands:

- `pwd` (print working directory)
- `cd ..` (change directory)
- `pwd`
- `cd .`
- `pwd`

What conclusions can you draw about the role of these directories?

- directory `..`
- directory `.`

2.2 The root

Go back to the highest level of the tree structure (either with multiple `cd ..`, either with `cd /`), then use the command `ls -ld`.

- What is the root called?
- Who is the owner of the root of the tree structure?
- What are your rights to this directory?

2.3 Jump from branch to branch

Go to the `/usr/bin` directory. (Command `cd`)

Jump to the `/usr/lib` directory (then return) using each of the two methods: *absolute naming* and *relative naming*.

“Have fun” doing the same thing by jumping from `/usr/share/man/man1` to `/usr/share/man/man2` and vice versa.

- In which cases might *absolute naming* be preferable, and in which cases might *relative naming* be preferable?

2.4 Your files and those of others (school computers)

- Return to your working directory: `cd`
- Go up to the parent directory: `cd ..`
- Check everything: `pwd` then `ls`
In principle, you should see a subdirectory with your name, your working directory.
- Try going to a colleague’s directory: `cd colleague’s_login`
- Check everything: `pwd` then `ls`
- Go back to the parent directory: `cd ..`
- Check everything: `pwd` then `ls`
Was it like that before? What happened?

Your files are not physically present on each of the school’s PCs. They are located in a storage array. Each PC can access them via the network. When a PC needs them, it mounts them, then after a certain period of inactivity, the files are unmounted. This is automatic and transparent. Typically, when you log on to a PC, it needs your files and mounts them. But if you request access to a colleague’s directory, it mounts that too. (Then, the fact that the colleague has set Unix permissions on their files is another story.) It’s a good way to share files easily (no need for a USB key or network transfer, it’s already on the network).

2.5 The meta-characters


Run the following command: `ls -l /bin/mk*`

- What do you notice?
- Deduce the role of the character `*`
- Now run the command `ls -ld /usr/bin/zi?` and compare it with `ls -ld /usr/bin/zi*`
- Deduce the role of the character `?`
- Compare, for example, `ls -ald .*` and `ls -ald .????*`

2.6 The autocompletion

Learn to type faster than your shadow!

The use of autocompletion depends heavily on the type of shell you have on your system (type `echo $SHELL` to find out).

- Enter the command `ls -l /bin/mk` followed immediately by the key <Tab>. The tab key is on the left side of the keyboard, and is often represented by . What do you see?
- Press <Tab> again and see the result. What help is provided?
- Complete the command with a few characters to remove any ambiguity and press the <Tab> key again to automatically display the name `/bin/mkdir`

Note that completion is a convenience feature offered by the shell you are using (`bash`, `tcsh`, etc.). With older shells such as `sh`, it does not work... With more modern shells, you can perform very subtle completion on command option names or arguments (for example, after `man`, you will only get completion on command names for which you have the manual online) or even spell checking.

2.7 Redirections

- Return to your working directory by simply typing: `cd`
- Ask for the time: `date`
What do you see?
- Now redirect the output of the previous command to the file `now.txt` in your home directory:
`date > now.txt`
What do you see?
- List the contents of your working directory: `ls -l`
So?
- Display the contents of this file: `cat now.txt`
Compare with the current date: `date`

2.8 Communication pipes

How many times have you typed the `cd` command since earlier?

No, don't try to remember, the computer has a better memory than you do.

- View your command interpreter history: `history` This includes both the contents of the hidden file `~/.bash_history` and the contents of its memory since you opened your terminal. When you close your terminal, this file `.bash_history` file is updated.

- Everything is displayed too quickly on the screen. Page through using the `more` command or the `less` command² in the following manner: `history | less`

You can scroll using the standard keys <Page Up> <Page Down>. In addition, for various historical reasons (terminals did not always have such sophisticated keyboards...), you can use the <Space> key to display page by page or the <Enter> key to display line by line. To exit, type <Q> (Quit).

- Okay, let's get back to it. We're not going to do the work by hand, but have the computer do it. That's what it's there for. Let's have it search for lines containing `cd`:

```
history | grep cd
```

- All that's left is to have it count the lines:

```
history | grep cd | wc -l
```

2.9 Foreground and background commands

- Run the command `xeyes`. In the same terminal, now run any command (e.g., `pwd`). What do you see?
- Press <Ctrl-C> to kill the `xeyes` command and restart it in the background (find out how to do this either in the course or on the web page mentioned above). What difference do you notice? What can you do now that you couldn't do when the command was running in the foreground?

3 Unix commands documentation

Remember to RTFM: *Read That Fine Manual*. Each Unix command is documented in the online Unix manual and accessible via the `man` command.

As soon as you have a problem with a command, do: `man command_name`

You can search for a term in the manual page by typing `/word_to_search`, which indicates the first occurrence of the term. To view subsequent occurrences, use the `n` (*next*) key. To exit the manual page, use `q` (*quit*). (In fact, these keys are managed by your *pager*, typically the `more` or `less` command, and you can read the manual for more information...).

Each piece of documentation is structured in the same way. You will mainly find:

- the name of the command and its short description;
- a synopsis, summarizing how to use the command;
- a more complete description of how the command works, with often examples of use;
- the list of accepted options, with their meanings;
- at the end of the manual page, the SEE ALSO section can guide you to other related commands.

²The world of Unix is a world of humor, especially when it comes to command names . A humor that is sometimes a little elitist...

In general, take the time and trouble to consult these manual pages. (I insist on this because that's how you learn a lot!)

Using the reference manual, answer the following questions:

- How do you make a recursive copy of files and directories (command `cp`)?
- How do you display the contents of a directory in reverse alphanumeric order (command `ls`)?
- What are the options for the `man` command, and more specifically, what does the `-k` option do?
- Which pages of the manual deal with the word *listing*?
- Which pages of the manual deal with the word *image*?

Note also that the command interpreter (the shell) also includes a few internal commands (`cd` `pwd` `..`). You will therefore find explanations of these commands in the `man` page for your shell. In addition, some modern shells can provide you with online help for their internal commands. For example, in `bash`, you can type `help cd`.

4 Mini-exercise: preparing the DCL workspace, using only the terminal

If you haven't already done so, it's time to prepare your workspace for the TAF DCL and for each UE using only the terminal. Ideally, minimize the number of commands used. You will use the following commands: `cd`, `ls`, `pwd`, `mkdir`, `rmdir`, `mv`, and `rm`.

Proposed file organization (visualized using the `tree`) command:

```
DCL/
├── CONC
│   ├── miniproject
│   └── tp
├── ECO
├── IDL
│   ├── project
│   └── tp
├── MAPD
│   ├── petrinetproject
│   └── tp
└── adm
```

5 Configuring your work environment in terminal mode (on school computers)

Your environment is already configured to allow you to use most standard commands. However, if you want to use a non-standard command (e.g., for a specific lab exercise, or because it is a command that you have installed yourself), you will need to configure a number of things in your environment.

The most important thing is to know where this command is located. So we will set a variable in your environment so that the shell can find this command without any problems. The variable in question is `PATH` (in uppercase) and contains the list of directories where the commands that can be called by their simple name are located. To view its contents, use the command `echo $PATH`. Note that this variable `PATH` may be empty, for example if your shell was installed (and compiled) with a default path such as `/bin:/usr/bin`.

5.1 The `SETUP` command (specific command of the school environment)

The environmental modification to be made is complex for the novice, so at school, our system administrators have developed a special command called `SETUP` (in uppercase) to make things easier.

- Run this command to list the available software.
- Display the *path* known to your environment for searching for user commands: `echo $PATH`
- Run the command `java -version` and locate the default version of your Java.
- Use the `SETUP` command to request a version change
- Using the `SETUP` command, request a change of version (e.g., `SETUP JAVA17` or `SETUP JAVA18`)
- Verify that you are now working with a different version of Java: `java -version`
- Check the changes in your configuration: `echo $PATH`

Please note that the `SETUP` command adds but does not remove... For example, if in the same terminal you run `SETUP JAVA17` then `SETUP JAVA18`, you will end up with an environment configured to search *first* for version 1.7 of Java, then version 1.8... and will therefore always find Java 1.7 first! (To see for yourself, type `java -version` or `echo $PATH`.)

That said, there's nothing stopping you from typing `SETUP JAVA17` in one terminal and `SETUP JAVA18` in another...