Introduction to Linux & containerization
-
Practical session

0.1 - Goals

The purpose of this practical session is to discover and use some common Linux shell commands to manipulate files in a minimal Linux environment. If you ever need to work with High-Performance Computing (HPC), this environment closely resembles what you will encounter (with additional tools and functionalities). Familiarity with these tools can save you time in the future.

0.2 - Context

For this lab, we will use the XTB software (Semiempirical Extended Tight-Binding), which implements the algorithms described in this paper. XTB provides geometric optimization functions for molecules. We will use these functions to determine the optimal geometry of water, ethanol, and caffeine molecules from very noisy versions of them.

In the second part, we will perform the same calculations using a different compiler and a different computational library to assess whether these differences lead to notable variations in the results and computation time.

0.3 - Virtual Machine Configuration

By default, the virtual machine (VM) is configured to use a single CPU core, but you can allocate more. To do so, go to the VM settings, select the "Processor" tab, and increase the number of cores. Allocating 4 cores should be sufficient.

Once the VM is launched, log in using the following credentials:

- **Login:** student
- **Password:** student

The VM is set to use a QWERTY-US keyboard layout. To switch to a French layout, use the following command:

```
loadkeys fr
```

Next, you will need to install Git. To do this, switch to the root user and install Git using apt (the root password is also "root"):

```
su root
```

```
apt install -y git
[ ... apt running ...]
exit
```

1.1 - Retrieving the Files

After installing Git, clone the following repository to download the necessary files https://github.com/DIADEM-Summer-School/2025-school-resources.git

You will find the *02_Linux-Containers* folder, which contains the lecture slides and a practical-session directory that will serve as your working directory for the rest of the lab.

1.2 - Creating the Container

The `xtb.def` file is an Apptainer image definition file that compiles XTB using Intel's C and Fortran compilers, as well as their Math Kernel Library (MKL). You can view its contents using `cat` or `less` to understand its structure. Pay particular attention to the `%runscript` section, which defines the program executed when the `run` command is used.

Once you have reviewed the file, build the image using Apptainer's build function and name it `xtb-intel.sif`. Note that the Intel libraries are quite large, so the build process may take some time depending on your internet connection.

1.3 - Using xtb

The XTB documentation is available here. Focus on the "Geometry Optimization" section, where you will find an example command to run an optimization with default parameters.

Start by optimizing the water molecule. Remember that you are running XTB through an Apptainer container, so the initial command must call the container.

1.4 - Processing the Data

After the optimization completes, use the `ls` command to list the new files generated. Not all files are relevant, so you should organize and filter them for easier access later.

Begin by creating a `results-intel` directory. Since we aim to compare two versions of XTB, specifying the version is important.

For this experiment, you should save the following:

- The optimized molecule's `.xyz` file.
- The energy of the optimized molecule (extracted from the `.xyz` file).
- The number of optimization cycles.
- The total execution time.

The first two items are straightforward to retrieve. The latter two are more challenging, as

they are not saved in the output files but are only displayed during execution. You will need to find a way to capture and process this information programmatically.

Note: In this basic scenario, you could manually observe the number of cycles and execution time. However, in real-world usage, where you might process many molecules or complex molecules requiring hours of computation, manual observation is inefficient. Automating this process is essential.

Once you have successfully saved the required information, repeat the process for ethanol and caffeine.

2.1 - Building a New Image

The next step is to use open-source compilers and computational libraries. The XTB documentation provides guidance on compiling the source code. Start by editing the `xtb.def` file with a text editor like `nano` or `vim`. Comment out the "Intel compiler" section and uncomment and complete the "Gnu compiler" section.

Update the compilation variables as specified in the documentation.

Note: *GNU compilers use standard installation directories, so the system automatically checks these locations during compilation and execution. Unlike with Intel, you do not need to specify library paths using:*

```
export LD_LIBRARY_PATH="..."
```

After editing the file, rebuild the image and name it `xtb-gnu.sif` to distinguish it from the first image. If the build succeeds, repeat steps 1.3 and 1.4 using the new image (create a new results directory). Compare the calculations and execution times between the two compilation methods for the test molecules.

2.2 - Open-Ended Question

Is this comparison between the two methods scientifically valid? If not, how could it be improved to ensure scientific rigor?