Working Sessions On Linux

The document provides a basic knowledge of working with a Linux computer. Emphasis is given to describe some basic linux commands to get along with the computational environment needed for most of softwares used in quantum chemistry.

The document is largely inspired from Rémi Marchal's introduction to Unix (Western Pole of the French research federation TheMoSiA training, Feb. 2022).

Contents

1 Introduction

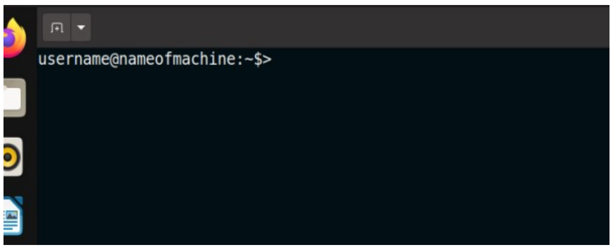
Unix/Linux is a family of multi-tasking and multi-user operating systems (OS), deriving from the original Unix OS created in the seventies by Ken Thompson.These OS are quite popular in the academic community, and it may be noted that all scientific supercomputers are using Unix (mainly Linux-based OSs, but some also rely on MacOS). These OS all share common ideas, noticeably the concept of command-line interpreter (usually called shell). **In this document, we will mainly focus on the Bourne-Again Shell (bash), common to all Linux distributions.**

2 Basic commands

In recent Linux distributions (for example ubuntu https://www.ubuntu-fr.org/), a significant effort has been dedicated to provide user-friendly interfaces, allowing to operate and even manage the OS without the need to resort to the terminal. Nevertheless, in scientific modelling it is often necessary to invoke at some point a terminal, to run some specific commands. We propose a brief introduction of working with a linux computer to get along with the computational chemistry environment. Of course, more detailed linux introductions can be found on the web, for example here : Ubuntu tutorial

2.1 The shell

When opening a Linux terminal (running under bash), the user will usually observe something like :



```
username@nameofmachine:~$>
```

A linux terminal is displayed on the previous snapshot. For this example, username is the login of the active user and nameofmachine is, as expected, the name of the machine. Overall, the syntax to invoke a command in the terminal :

```
command -[variations] argument_1 argument_2 ...
```

where `[variations]` are options that the command may accept (for instance `-r` for recursive operations), and arguments are to be passed (often, an input and an output). In case of doubt on the actual syntax of a command, one may look into its manual, by invoking

```
man command
```

2.2 Things that do not work in a terminal, and how to get away from the problems these may create

1. Ctrl+S will not save the content of a terminal, but rather freeze it completely. Ctrl+Q will unfreeze it.

2. If at some point your terminal is not responding but was not frozen by a Ctrl+S, you may interrupt the current task by using Ctrl+C.

3. The `exit` command allows to close a terminal; this is noticeably the command to use to stop a ssh connection.

2.3 How to travel through the files tree ?

In linux OSs, files are structured according to a so-called file tree: files have a given location in a sequence of directories, starting from the root of the tree: "/".

In a terminal, the `pwd` command returns the present location in the file tree. The location given out by `pwd` right after logging is named the user's home folder:

```
username@nameofmachine:~$ pwd
/home/username/
```

→ `/home/username/` is the home folder of "username".

Locations in the file tree can be defined in two ways: relative or absolute. Say for instance that the home folder contains 2 directories: dir1 and dir2. The addresses of these two directories are

- in absolute position:

  ```
  /home/username/dir1
  ```

  and

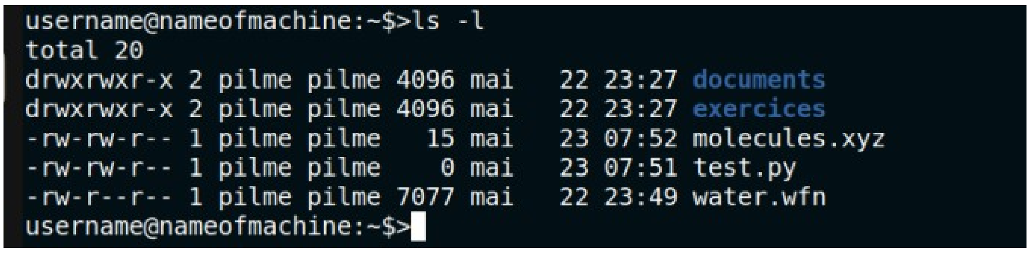
  ```
  /home/username/dir2
  ```

- relatively to the home folder : `dir1/` and `dir2/`

The command cd

Also known as 'change directory', the command cd is used to change the current working directory. It must be followed by the (relative or absolute) location of the directory. Some useful variations on locations:

1. `cd ..` asks to access the directory immediately above in the file tree;
2. the previous command can be repeated, or alternatively one may type `cd ../..`
3. `cd -` ask to return to the previous active folder in the terminal ;
4. `.` means "active folder"
5. ~/ means "home folder".



```
username@nameofmachine:~$>ls -l
total 20
drwxrwxr-x 2 pilme pilme 4096 mai   22 23:27 documents
drwxrwxr-x 2 pilme pilme 4096 mai   22 23:27 exercices
-rw-rw-r-- 1 pilme pilme   15 mai   23 07:52 molecules.xyz
-rw-rw-r-- 1 pilme pilme    0 mai   23 07:51 test.py
-rw-r--r-- 1 pilme pilme 7077 mai   22 23:49 water.wfn
username@nameofmachine:~$>
```

2.4 Files and folders manipulation

1. The command ls
   `ls` allows to list the content of the active folder (or the content of any folder if its adress is passed as an argument). `ls -l` allows to list with details, noticeably regarding the permissions, as shown on the example below:

   In the output above, the first column lists the permissions[a], then the user and groups owning the file are given, then the file size in bytes, the date of its last modification, and its name.

   To order the file by date of last modification, user can use `ls -lt`, and this order can be reversed by `ls -lrt`.

2. The command mkdir
   To create a folder, one may use the `mkdir` command, giving the address of the target directory as argument.

3. The command cp
   The command to copy a file is `cp`. It requires two arguments:

   ```
   > cp file copied_file
   ```

   the first one is the file to be copied, and the second is the target of the copy (copied_file). Of course, absolute and/or relative addresses can be provided here. The same command can be used to copy a directory and its content, provided that a `-r` option is provided.

4. The command mv
   Moving a file can be done by the `mv` command. As cp, it requires two arguments: the address of the file to be moved, and the target address. If the target is located in the same directory, `mv` falls back to a simple renaming. Contrary to cp, mv does not need the `-r` option to displace a folder.

5. The command rm
   Removing a file can be done by `rm`, followed by the list of files to remove. **Be careful that this operation is permanent: a removed file cannot be recovered.** Removing folders and their content can be done with `rm -r`.

   Finally, `*` can be used as a wildcard for any character string; for instance moving all files from the active directory to another folder can be done by `mv * /target/folder/`. To be used with caution, and advisably **never** in conjunction with rm.

[a]Permissions are given in the following order: user, group, and all other users. The letters r, w and x respectively mean "reading", "writing", and "executing". In the shown example, user "username" is allowed to read and write in the three files; all users in the "staff" group have the same permissions, and all other users can only read the files.

2.5 Miscellany

2.5.1 Redirecting outputs

Many commands naturally return text in the terminal; all this text can be redirected to a file following

```
command [arguments if any] > file
```

For instance,

```
ls *.xyz > listofxyz.txt
```

will produce a file named `listofxyz.txt` containing the list of all file with an `xyz` extension (one file per line, in alphabetical order).

2.5.2 Getting data

We describe here some commands that can be used to analyze a text file such as an output file of Gaussian or ADF. To illustrate these commands, let's imagine that you have a file called "output.txt" that contains the following (boring) text:

```
1 This is line1
2 is the second line
3 starts to be boring
4
5 eh, where is line 4?
6 how far is this going?
7 number of days in a week
8 are we tere yet?
9 are we there yet? (correct the typo in line 8)
10 We are done !
11 hum...
12 I thought we were done?
13 yes, but I need more than 10 lines in the examples
14 arg.
```

1. The command cat
   `cat file` will print in the terminal the content of the file. For example:

   ```
   $ cat output.txt
    1 This is line1
    2 is the second line
    3 starts to be boring
    4
    5 eh, where is line 4?
    6 how far is this going?
    7 number of days in a week
    8 are we tere yet?
    9 are we there yet? (correct the typo in line 8)
    10 We are done !
    11 hum...
    12 I thought we were done?
    13 yes, but I need more than 10 lines in the examples
    14 arg.
   ```

`cat file > file2` will print the content of file into file2. Note that several files can be printed this way: `cat file1 file2 file3 ... > fileN` will produce a file fileN which is the mere concatenation of file1, file2...

2. The command tail
   `tail file` will only print the last 10 lines of the file. For example:

```
$ tail output.txt
 5 eh, where is line 4?
 6 how far is this going?
 7 number of days in a week
 8 are we tere yet?
 9 are we there yet? (correct the typo in line 8)
 10 We are done !
 11 hum...
 12 I thought we were done?
 13 yes, but I need more than 10 lines in the examples
 14 arg.
```

   `tail -X file` will print the final X lines of a file. For example:

```
$ tail -5 output.txt
 10 We are done !
 11 hum...
 12 I thought we were done?
 13 yes, but I need more than 10 lines in the examples
 14 arg.
```

   `tail -f file` will stream the end of the file in the terminal; this can be helpful to monitor the fate of a calculation, if the program you use regularly writes in a text file. Downside: it holds the terminal, so an interruption of task needs to be done (ctrl+C) to provide new commands.

3. The command head
   `head` is a complementary tool to tail: it will print the first 10 lines (or X with the -X option) of the file. For example:

```
$ head output.txt
 1 This is line1
 2 is the second line
 3 starts to be boring
 4
 5 eh, where is line 4?
 6 how far is this going?
 7 number of days in a week
 8 are we tere yet?
 9 are we there yet? (correct the typo in line 8)
 10 We are done !
pa3059fl@krenek02 CentOS_Intel64_7.7 [/user1/icmub/pa3059fl/tmp]
$ head -5 output.txt
 1 This is line1
 2 is the second line
 3 starts to be boring
 4
 5 eh, where is line 4?
```

4. The command grep
   `grep "stringtolookfor" file` will print in the terminal all occurrences of the string "stringtolookfor" in file. For example:

   ```
   $ grep "done" output.txt
    10 We are done !
    12 I thought we were done?
   ```

   This can be very helpful if your file is very long and you are only interested in a single value, or in conjunction with wildcards if you have many files in folder and want the same data for each (for instance the final SCF energy for a set of molecules).

5. The command tar
   The Linux 'tar' stands for tape archive (file.tar). It is used to create and to extract compressed data files.

   > tar [option] [file.tar] [file or directory to be archived/compressed]

   ```
   Some options :
   -c : Creates Archive
   -x : Extract the archive
   -f : creates archive with given filename
   -t : displays or lists files in archived file
   ```

2.5.3 Running program

To execute an executable (binary) from a command-line, simply provide its address:

```
/path/to/executable
```

or

```
./executable
```

if it is located in the active folder. If this executable prints in the terminal, you can redirect the printing to a file by

```
./executable [eventual arguments] > outfile
```

3 Some tools needed for the working sessions

Some specific linux commands or softwares will be needed for the sessions. They can be started directly from a command-line or if applicable, by a click on the linux desktop. Here the main ones.

3.1 Gedit

gedit is a default text editor designed for the ubuntu environment. It emphasizes simplicity and ease of use, with a clean and simple Graphical User Interface (GUI). gedit could be designed for the reading of results (text files).

```
> gedit results.pop
```

3.2 Vi

The popular command-line text editor that comes with the LINUX/UNIX operating system is called vi (visual editor).

```
> vi results.pop
```

It has two modes of operation:

- *Command mode* which causes action to be taken on the file.
- *Insert mode* in which entered text is inserted into the file.

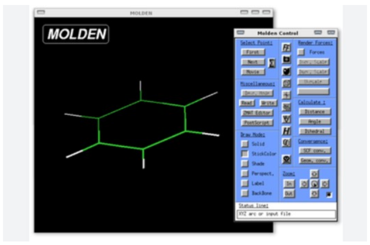
Many "Quick Start" tutorials are available on the web. For example, see : vi quick start

3.3 Molden

Molden is a general free package for displaying molecular and electronic structure processing program from standard quantum chemistry programs such as the Gaussian log file.

To start molden in command-line:

```
> molden -l molecule.xyz
```



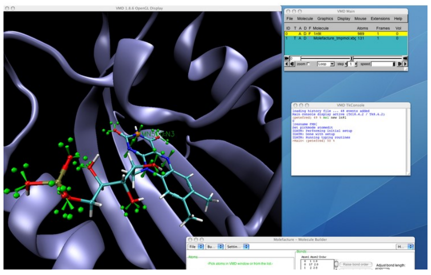
Here "molecules.xyz" is the file containing the cartesian coordinates of atoms in the molecule (XYZ format). Details of the format can be found here: xyz format.

3.4 Vmd

VMD is a free molecular visualization program for displaying, animating, and analyzing molecular systems using 3-D graphics and built-in scripting. It can be used to visualize both cube files and xyz files.

Free Download from: https://www.ks.uiuc.edu/Research/vmd/

For example, an introductive tutorial: Visualizing your results with VMD



To start vmd in command-line:

```
> vmd -e file.vmd
```

Here "file.vmd" is a vmd visualization state file, which is directly produced by TopChem2 as an output file.

3.5 Gaussian

The Gaussian package is a general purpose computational chemistry software. The name originates from Pople's use of Gaussian orbitals. It solves complex quantum chemistry problems, predicting molecular structures, reactions, and spectroscopic properties. The current version available on the computers of the school is Gaussian 09. Gaussian will be available throughout all the MODERM sessions. It can be used to generate the needed wfn/wfx/cube input files for TopChem. To start a calculation in command-line :

```
> rung09 input output
```

3.5.1 How are generated the Gaussian wfn/wfx files ?

The output keyword needs to be added in the route section. It is used to create the wavefunction file (.wfn or .wfx). This latter popular format records occupied orbitals (virtual orbital can also be included) and their corresponding expansion coefficients with respect to unnormalized primitive cartesian gaussian basis functions.

In order to generate the gaussian wfn/wfx files, we need to set some spefications in the gaussian input file. Set out=wfn in the route section and give the name of the wfn file at the end of the input.

- For example, an input for a DFT calculation,

  ```
  #P B3LYP/6-31G(d,p) opt out=wfn #(or out=wfx)

  CO

  0 1
  C
  O 1 1.14

  co.wfn
  ```

  To obtain the conceptual DFT descriptors (within the FMO approximation), we need to print some virtual orbitals in the wfn file, at least the LUMO.

  In the Gaussian route section, add :

  ```
  iop(99/18=Nvirtual)
  ```

  Nvirtual is the desired number of virtual orbitals (integer). For a non degenerated LUMO only, set Nvirtual to 1.

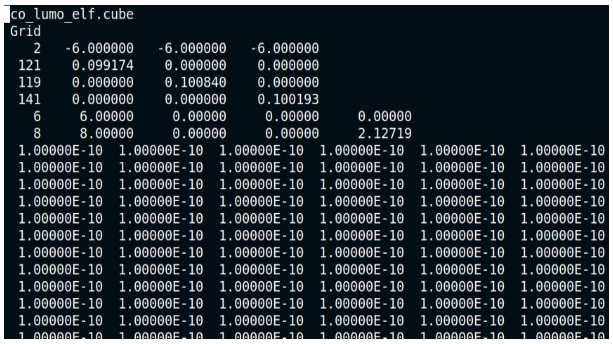
- Post-Hartree Fock calculations. The NaturalOrbitals should be specified as an option in the route section for the population keyword and all in the density one.

  ```
  #P MP2/6-31G** opt pop=no density=all out=wfn
  ```

3.5.2 Gaussian cube files

The cube format (text file) describes a scalar field numerically represented by its values distributed on a suitable 3d uniform grid of N points with a constant spacing between the point. It also contains atom positions. The used scalar field can typically be, the electron density, the electron localization function (ELF), the non-covalent interactions index (NCI) or the Molecular Electrostatic Potential (MESP).

The format originates from the Gaussian software but currently, most of quantum chemistry programs produce cube files. The file contains of a header which includes the atom details and the size as well as orientation of the voxel data. This is followed by all volumetric data, one scalar per voxel element. The file is divided into two parts: a header part and a section for the volumetic data.



```
co_lumo_elf.cube
Grid
   2   -6.000000   -6.000000   -6.000000
 121    0.099174    0.000000    0.000000
 119    0.000000    0.100840    0.000000
 141    0.000000    0.000000    0.100193
   6     6.00000     0.00000     0.00000     0.00000
   8     8.00000     0.00000     0.00000     2.12719
 1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10
 1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10
 1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10
 1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10
 1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10
 1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10
 1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10
 1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10
 1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10
 1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10
 1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10
```

== **Header part** ==

- The first two lines are comments and are generally ignored by most of packages. The third line gives the number of atoms (2 for the above example) followed by the position of the origin of the volumetric data, the values are given in Bohr by default.
- The next three lines provide the number of voxel data along each cartesian axis (Nx, Ny, Nz) followed by the axis vector. The length of each vector (here 0.1 bohr) is the length of the side of the voxel, non cubic systems are allowed. A suitable length for a proper analysis of a cube file is 0.1 bohr.
- In the last section, each line includes the atom with its corresponding atomic number (6 and 8), its nuclear charge, and the three (x, y, z) coordinates indicating its location.

== **Volumetric data** ==

The voxel data are provided for each (x,y, z) grid point according to a typical standard scheme with the x axis as the outer loop and the z axis as the inner loop, for example, written as

```
do ix=0, NX
   do iy=0, NY
      do iz=0,NZ
      .....
```

How can these files be generated ?

- TopChem2 produces standard cube files for save the scalar field data. It also uses cube files as input files.
- ADF can also produce cube files, see here.
- It is also possible to produce cube files using some gaussian utilities which are normally provided with the gaussian package. For example, the cubegen utility generates a cube file from a simple command-line,

  ```
  cubegen 1 density=scf test.fchk test.cube 120 h
  ```

  The above command-line produces a cube file named test.cube (with header) from a fchk file named test.fchk. The cube file contains the SCF electron density computed inside a rectangular 3d-grid of 120x120x120 points.

  ```
  cubegen 1 potential=scf test.fchk test.cube 80 h
  ```

  The above command-line produces a cube file named test.cube (with header) from a fchk file named test.fchk. The cube file contains the SCF electrostatic potential computed inside a rectangular 3d-grid of 80x80x80 points.
- The VASP output files (CHGCAR and ELFCAR) can be easily handled or converted to cube files using free scripts, here.

How can these files be visualized ?

The 3d data can be viewed as isosurfaces[b] using a standard molecular visualization program such as VMD. Some details and tutorials can be found here : View Cube with VMD or Visualizualing your results or in the TopChem2 manual.

[b] An isosurface is a three-dimensional representation of a scalar field, defined by a specific threshold value, used to visually highlight regions of interest within the data.