Einführung in die Programmierung für Physiker

SoSe 2024 – Marc Wagner

Michael Eichberg: `eichberg@itp.uni-frankfurt.de`

Exercise sheet 1

To be handed in on 26.04.2024.

Exercise 1

Installing GNU/Linux Having a GNU/Linux system at hand is probably a very good idea in this course. It is flexible and powerful, and its most fashionable distributions (Fedora, Debian, Ubuntu, Suse...) bring by default many tools for programming. And provide easy access to much more software through their package managers.

This first exercise is a reminder of what was said in the lecture. The simplest GNU/Linux installation is probably done on a clean computer - without any other operating system. But it is usually possible to make GNU/Linux coexisting with other operating systems. Many distributions have online documentation to help you in such a task.

Another option is to use some kind of virtualization software. In that case, you install some program like VirtualBox or Xen (or many others) in your current operating system. The general idea is that those programs simulate computers, therefore you can create a new virtual computer inside the virtualization program and install whatever operating system you like without affecting the installation of the current real operating system.

You are free to choose the method you like more.

Exercise 2 [*Playing with the shell*]`(1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 8 pts.)`

The following exercise can be submitted as a series of screenshots of the terminal with explaining text in one PDF.

(i) Open a terminal and execute the command `cd`. In which folder are you? Use `cd`, to navigate to the folder `Documents` (or `Dokumente`) and create the sub-folders `ExerciseSheets` and `ExerciseSheet_01`, ...`ExerciseSheet_12`. In one line (e.g. using globs/wild cards), move all directories `ExerciseSheet_XX` to `ExerciseSheets/`. What happens, if you omit the slash `/` at the end? Confirm that everything is in the correct place by showing the content of your current directory and `ExerciseSheets` via `ls`.

Hint: If the folder `Documents` (or `Dokumente`) does not exist yet, create it using `mkdir`.

(ii) Explain the difference between the output of `ls ExerciseSheets` and `ls $(pwd)/ExerciseSheets`.

Hint: Consider, what `pwd` does. You can check with `echo "<ls-input>"`, how the `ls` command reads the input.

Produced with the ExerciseHandler

(iii) Create two directories `.BonusSheet` and `.ExtraSheet` and execute `ls`. Can you see the new directories? Add the required option for `ls` to show these directories. What are the directories `.` and `..` shown?

Hint: See `man ls`.

(iv) Check if `.BonusSheet` and `.ExtraSheet` exist in your present directory and remove them using `rm *Sheet`. Does it work? What happens for `rm .*Sheet`? Remove the folders using either `rmdir .*Sheet` or `rm -r .*Sheet` and explain the difference between the two options.

(v) For the following tasks, enter the directory `ExerciseSheets/ExerciseSheet_01`. Execute `ls ..` and redirect (>) the output to the file `contents.txt`. Make a copy `contents_copy.txt` of `contents.txt` and append (>>) the output of `echo "Hey, what's up?"` to `contents.txt`. What does `diff contents.txt contents_copy.txt` show?

(vi) Enter the directory `ExerciseSheets/ExerciseSheet_01` using `cd` and create a sub-directory `scripts`, along with a file `scripts/test.bash`. You can check the global path of your current directory using `pwd`. What is the global path of the file `test.bash`, that you just created?

(vii) For the following tasks, enter the directory `scripts`. Open `test.bash`, insert the line `echo "You are in directory $(pwd)"` and execute `./text.bash`. Do you have the required permissions? Check the permissions of the file via `ls -l`.

(viii) Change the permissions of the file, such that only you can write to the file and everyone can read and execute the file. The output of `ls -l` should look like

```
-rwxr-xr-x 1 <name> <name>     0 <month> <day> <time> test.bash
```

Exercise 3 [*Gnuplot*] (3 + 1 + 2 = 6 pts.)

(i) Open Gnuplot by typing gnuplot in the terminal. Use `help plot` to access information of Gnuplot's `plot` command. Draw a curve by executing `plot sin(x)*x**2`.

Remark: Gnuplot has a set of standard functions you can use when plotting, some of which are summarised below:

```
exp()   log()   **
sin()   cos()   tan()
asin()  acos()  atan()
cosh()  sinh()  tanh()
```

Plot the Gaussian

$$\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}\,.$$

Configure the plot with the following settings:

Produced with the ExerciseHandler

(a) Plot labels x and y,

(b) Plot ranges $x \in [-5, 5]$ and $y \in [0, 1/2]$,

(c) Tics on x-axis in steps of 1, tics on y-axis in steps of 0.1,

(d) Special named tics at $x = -1.96$ and $x = 1.96$ named "95%",

(e) Choose a plot title, e.g. "Gaussian distribution ($\langle x \rangle = 0$, var $= 1$)",

(f) A sample rate which results in a smooth plot.

This can be achieved by using Gnuplot's `set [option]` function. For additional information, see `help set [option]` for options

```
xlabel    xrange    xtics    title    samples    key
```

and use the `replot` function (instead of `plot`) to display changes.

Change Gnuplot's `terminal` and `output` to save the plot in a file of a given format (e.g. gif, png, jpg, pdf, tex, etc.). After that, close Gnuplot.

(ii) Download the file (e.g. using `wget`)

```
https://itp.uni-frankfurt.de/~eichberg/pprog-sose2024/
                    gnuplot_data.txt
```

The file contains three columns, corresponding to x-coordinate, y-coordinate and the uncertainty (error) of the y-coordinate. In Gnuplot, using

```
plot "data" using 1:2:3 with yerrorbars
```

plot the content of the file. What does `using 1:2:3` specify? How can you switch x and y coordinates? How can you can you plot the data without error bars?

Guess the function, which describes the data and check using `replot <function>`. Save the new plot.

(iii) Inform yourself (Gnuplot documentation/internet) about fitting data with Gnuplot and fit a suitable ansatz to your data. What are the resulting fit parameters?

Exercise 4 [*Homework zipper*] (1 + 3 + 1 + 1 = 6 pts.)

(i) Create an executable Bash script `ZipHomework.bash`, where the first line is either the shebang

```
#!/bin/bash
```

or

```
#!/usr/bin/env bash
```

Produced with the ExerciseHandler

Remark: The details of the shebang are not relevant for now.

(ii) Define a variable with your name, e.g. `"<Lastname><Firstname>"`. Let your script take one positional argument `<dir>`, such that you can execute it as `./ZipExercise.bash <dir>`. `<dir>` is supposed to be a directory name (e.g. `ExerciseSheet_XX`). Inside the script, you can use positional arguments as `$1`, `$2`, ... Let the script execute the following steps:

   (a) Rename of the directory from `<dir>` to `<name>_<dir>`,

   (b) Execute `tar -czvf "<name>_<dir>.tar.gz" "<name>_<dir>/"`,

   (c) Change the directory name back to `<dir>`.

   You can add more commands to your scripts, e.g. `echo`, to print variable values, or a message of what the script is currently doing.

   Remark: It is good practice, to wrap strings in quotation marks `""` or `''` (there is a difference between the kind of quotation mark). Without quotation marks, Bash treats blanks, whitespace and newlines as seperator between words. It is also good practice to combine `$` with braces, i.e. `{}`, to avoid ambiguities in strings like `"${...}...`. Mind, however, that this has a completely different meaning than `$()`.

(iii) What is the purpose of the `tar` option sequence `-czvf`?

(iv) In one line, how can you extract the zipped archive? What is the name of the archived directory?

From now on, you can use this script to zip your homework. Your tutor will be happy, if they only have one file per student to download and having a unified naming scheme ;)

Produced with the ExerciseHandler