ACADEMIS

First Steps on the Linux Command Line

tutorial

Table of Contents

First steps on the Linux Command Line

What this tutorial is about?

This tutorial lets you learn the basics of the Linux command line. You can learn commands to navigate directories, manipulate files and start other programs. If you have no previous experience with Unix-like systems or know a few commands but would like to know more, this tutorial is for you.

Prerequisites

This tutorial was prepared for Ubuntu Linux, but it works on MacOS, Cygwin and the Git bash as well, given that Python 3 is installed on your system.

Preparations

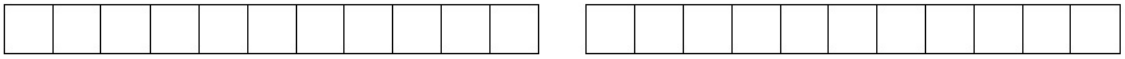
- Copy the file **Exercises.zip** from https://github.com/krother/Linux_Commandline_Tutorial/raw/master/Exercises.zip) to a computer with Ubuntu (or some other Linux) installed.
- Unzip the file.
- Type:

  chmod -R a+x unix_tutorial/ chmod -R a-x unix_tutorial/exercise6/check_permissions

- Explain trainees how to open a Unix shell
- Make this tutorial and a ‘Unix/Linux Command Reference’ document available (see PDFs in exercise material).

Your Task

In this tutorial, you will be looking for a word with 22 characters. All characters are hidden in the exercises below. All exercises can be solved using the Unix command line.



License

1. Directories and files

1.1. Navigating directories

The **first character** is hidden in a file somewhere in the *exercise1* directory tree. Use the commands

```
cd <directory_name>
```

(do not type the pointy brackets, just insert the directory name) and

```
ls
```

to move from one directory to the next. Look through subdirectories until you find one with the name *solution_1.1* and list its contents. If you went to a wrong directory, you can go back one level by typing:

```
cd ..
```

or to go back to the beginning:

```
cd
```

1.2. Show a hidden file

Some files are not visible immediately. To see them, you need the command

```
ls -a
```

The **second character**, is in the same directory as the first one, but in a hidden file.

1.3. Execute a program

Use *cd ..* to go back to the directory *exercise_1/directoryB/*. When listing its contents, you should see a program file. To find the **third character**, you need to execute the program. In Unix, this is done by typing

```
./program_name
```

1.4. Find out how big a file is

Go to the *exercise_1/directoryC/* catalog. To find **the fourth character**, you need to find out how big the text file in the directory is. This is done with the command

```
ls -l
```

In the table the command produces, you will find the file size in bytes, the file's owner, permissions to read and modify it, and the date/time of the last modification. When you want to obtain the fourth character, type

```
./file_size_check
```

The program will ask you for a file size.

Hint

When typing names of directories or files, try typing the first three characters, and press <*TAB*>. Unix tries to guess what you are typing.

2. Edit text files

Please use *cd ..* to go back to the top directory of the tutorial material. Then, change to the directory *exercise_2*.

2.1. See what is in a text file

In the directory *exercise_2/*, you will find a text file *solution_2.1.txt*. The **fifth character** is inside that file. To see its contents, use the command

```
more <filename>
```

2.2. Edit text files

To get **character number six**, you will need to create a text file in the *exercise_2* directory. On Ubuntu, you can do this using the editor *gedit*. You can start it typing the name of the program, or

```
gedit <filename>
```

Create a text file with a name of your choice in the *exercise_2* directory that contains the phrase

```
Please give me solution 2.2
```

Afterwards, run the program

```
./text_file_check
```

Hint

If you want to know more about a particular command, type

```
man <command>
```

You get shown a help page that you can leave by pressing 'q'.

3. Copy and remove files

Please go to the directory *exercise_3*.

3.1. Create a directory and copy a file to it.

To find **characters seven and eight**, you need to create a subdirectory named *solution* in *exercise_3/* and copy the file *code.txt* to it. For creating directories, you can use the command

```
mkdir <directory name>
```

For copying, you can use the command

```
cp <filename from> <filename to>
```

Use *ls solution/* afterwards to see whether the file is there. The program

```
./check_code
```

will give you the solution.

3.2. Removing files

In the same directory, there is a file *junk.txt* that does not contain anything useful, and should be deleted. To do so, use the command

```
rm <filename>
```

Also, there are more files to be deleted in the *data* directory. To remove more than one file at once, you can use '*' as a wildcard, i.e. 'rm pro*' will delete all of *profile, 'protein.txt and prototype.doc*. To get **characters nine and ten**, please run the program

```
./check_junk
```

after removing the files.

WARNING

On Unix, it is not possible to undelete files!

Hint

To remove an empty directory, you can use

```
rmdir <directory name>
```

To delete a directory and everything in it, the command *rm -r* exists. In combination with the '*' symbol, this is dangerous, because it gives you the possibility to wipe out all your data with a single command (e.g. if you type the wrong directory by accident). Thus, first make a backup and then play with this command.

4. Process text data

Please go to the directory *exercise_4*.

4.1. comparing two files

There are two different versions of a text, *foo_long.txt*, and *foo_short.txt*. To find out, how they differ, Unix provides the command

```
diff <filename1> <filename2>
```

Of course, you can look at the text first using *more* or a text editor. The **11th character** of the solution is the first character of the third last word in the output of *diff*.

4.2. Sorting a text file

Unix has a small program to sort text files alphabetically. It is called by

```
more <filename> | sort
```

The symbol '|' is called a pipe and is often used to connect Unix programs to each other. The **12th character** of the solution is the first character of the last word of the first line from the alphabetically sorted file *10000_lines.txt*.

Hint

To store the sorted lines in a new file, you can add an output file, like

```
more <filename> | sort > result.txt
```

4.3. Finding words in a text file

To look for specific words in a text file, the

```
grep <word> <filename>
```

command should be used. It produces all lines from the given file that contain the given word. The *grep* command is very powerful and can handle fuzzy search patterns called Regular Expressions (not covered here). The **13th character** of the solution is the first character of the second word in a line containing *'fool'*.

Hint:

You can search through many files at once by including a '*' in the filename.

5. Unzip files

Please go to the directory *exercise_5*.

5.1. unzipping archives

Unzipping compressed files is a very basic and important task. On Unix, you often encounter WinZip archives, .tar archives, and .gz compressed files. For unpacking Win zip files, use

```
unzip <filename>
```

for .tar and .tar.gz files

```
tar -xf <filename>
```

and for .gz files,

```
gunzip <filename>
```

The **14th and 15th character** of the solution are in a multiply wrapped archive in the *exercise_5* directory.

Hint

To pack a directory and everything within, you can use the command

```
tar -cf backup.tar <directory>
```

To subsequently compress it, use

```
gzip backup.tar
```

6. Run useful programs

Please go to the directory *exercise_6*.

6.1. Changing file access rights

Each file on Unix has separate permissions for reading *'r'*, writing *'w'*, and executing *'x'*. When displaying them with

```
ls -l
```

there is one triplet of these permissions for the files owner, one for a group of users, and one for all others. The *chmod* command allows to change these permissions, e.g.

```
chmod a+x <filename>
```

grants all users the permission to execute a file, while *u-w* forbids the current user (oneself) to write to the file (thereby protecting it from being deleted accidentally). The **next two characters** of the solution will be shown when you execute the program

```
./check_permissions
```

Hint:

You can grant permissions for a whole directory tree using

```
chmod -R a+x <directory>'.
```

6.2. How much disk space have I left?

To find out, how much disk space you have left, you can use the command

```
df
```

It lists all hard disk partitions, CD-ROMs, pendrives etc. Your data is stored in */home/*, if it does not exist, in */*. All numbers are given in kilobyte (1000 byte or one 1000000th GB). To obtain the **18th character**, run the program

```
./check_disk
```

6.3. Set an environment variable

To install some programs, it is necessary to set so-called environment variables. These can be set using the command

```
export <variable-name>=<value>
```

But then the variable will only be present in the same console window where you typed the command. You can see all variables by the command

```
env
```

To obtain the **19th character**, you need to set the variable *SOLUTION* to the value *6.3*, and run the program

```
./check_variable
```

Hint:

If you want to have an environment variable to be automatically set for eachconsole window, write the export command to the file *.bashrc* in your home directory (it is a hidden file).

Hint:

The *env* command can be combined nicely with *grep*, e.g. to check your PATH variable, you can type:

```
env | grep PATH
```

6.4. Check whether you have internet

The easiest way to check from the Unix command line whether the internet connection works, is to send a request to a known server (e.g. www.academis.eu) using the command

```
ping <web address>
```

The command reports, how long a message takes back and forth to the given server. To interrupt the messages, press *Ctrl+C*. You can use the program

```
./check_ping
```

to get the **20th character**.

6.5. Managing processes

To see what programs are running on your machine, type

```
top
```

It displays you a list of all currently active programs. *Shift+P* sorts them by the CPU time they are using, *Shift+M* by the amount of memory they are using (if you don't see any program consuming lots of memory, start a web browser).

The **last two characters** of the solution are the first two characters of the second word in the highlighted bar containing the column labels.

Hint:

If you want to get rid of one of the programs you started (e.g. because it crashed), you can do so by typing

```
kill <pid>
```

You find the pid number in the first column of the *top* output. Of course, you may only interrupt your own programs, not those owned by *root*, the system administrator.

Acknowledgements

This tutorial was created for participants in a bioinformatics workshop organized by Janusz M. Bujnicki for the EURASNET consortium in 2010. I would like to thank the organizers and all course participants, as well as Joachim Jacob for reviewing the material. Further thanks go to the German Academic Exchange Service (DAAD) for financial support.

Special thanks go to Allegra Via for using this tutorial in our course over the years, and to Pedro Fernandes for suggesting publication via Gitbook.

Contact

Dr. Kristian Rother

krother@academis.eu

www.academis.eu

License