Ubuntu terminal basics

# What is this whole ‘terminal’?

These days everyone uses computers. That is hardly a surprise, they are helpful in every field and simple enough, that even a child can use them – and they do. But computers weren’t always sun and rainbows. Back in the dark, dark days known for our parents as “youth” using a computer was very different.

You look at the screen and nothing seems familiar. There are no icons, no arrow cursor, not even a wallpaper, just some text and black background. You needed a special training or a nerdy, pimpled guy with braces to use such a machine. Luckily we moved past that with the inventions graphical operating systems. But if you want to get serious about your IT studies prepare for the Back to the Future experience, because you need to understand how this text on the black background worked – meet your new friend, terminal.

Terminal is a program allowing for communication with the operating system. You type text commands into it, describing what you want to do and computer writes back, letting you know about the results. It can do all the stuff you know from Windows – show you files, edit them, run programs, you name it. You might ask ‘if it does the same stuff as Windows, why shouldn’t I use the latter?’, and that’s completely valid question. Short answer to that might be: “speed, simplicity, IT friendliness”.

I know, putting “simplicity” there is a bit contradictory to the introduction, so let me explain myself. All you need to master terminal is practice. The longer you use it the easier it gets, to the point where it will be simpler then Windows. Let’s say you want to find all the files on your computer containing the word “bee”. In Windows you need to adjust your configuration, go through many dialog options, et cetera. In Ubuntu terminal you will just type ‘ grep -r “bee” / ’ and you are done – few moments later you will get all the occurrences of the word “bee” you’ve got on your computer. Now let’s presume you’ve got a file movie.webm but you need to convert it to mp4 format. In Windows you need to go through web to find software for that, install it, go through dialog options… Ubuntu allows you to type do it by simply “ffmpeg -i movie.webm movie.mp4” and ffmpeg program will handle the rest (if you don’t have this program installing it is also just one, simple command). With experience most of the stuff will get simple – installing programs will be simple, using them will be simple and, if in need, looking for help will be simple. But more on that later.

Speed comes in many forms. You can do stuff faster then in graphical systems – as I explained earlier single terminal command can do the job of many Windows operations. That means you can achieve the same result in 30 seconds rather then 5 minutes. Many commands will also work much faster – for example file searches are much faster on Ubuntu then on Windows.

# Paths and moving around

In the terminal you are always in some directory. Directories are defined by paths. Let’s introduce some commands connected with the subject:

pwd – if you type this command into the terminal it will tell you your current location, the directory you are in.

ls – this command shows you all the files in the current directory.

cd – this command let’s you move between directories, changes your location.

Let’s say you typed ‘pwd’ and got ‘/usr/local/documents’. That means you are in the directory named ‘documents’, which is located in the directory ‘local’, which is located in the directory ‘usr’. You can also treat the first ‘/’ as a top most directory containing all the other directories.

Now you typed ‘ls’ and the answer was:

dir\_1 dir\_2 file\_1

That means your current directory contains directories ‘dir\_1’, ‘dir\_2’ and file ‘file\_1’. Coloring might be different in your case, but the point is directories are marked by different colors then other files.

If you want to move to the directory ‘dir\_1’ you can type ‘cd dir\_1’ into the command line. Then you can verify that you changed your location by typing ‘pwd’. If you want to go back to folder ‘documents’ from here you can type ‘cd ..’ ( two dots mean ‘directory containing current direcorty’, so ‘cd ..’ is basically ‘go one directory up’).

Files are recognized by their paths. Path in terminal is similar to the normal life definition of the word – it is a way from one place to another. There are two types of paths – relative and absolute. Absolute paths start with ‘/’ and they show the way from the top directory to the file. Relative paths don’t have ‘/’ at the start and they show the way from the current position to a file.

Absolute path for ‘file\_1’ (from the previous example) would be ‘/usr/local/documents/file\_1’. For relative path we need to know our current position, so we check it with ‘pwd’ command (for this example let’s assume we got ‘/usr/local’ as result of ‘pwd’ command). Now we know our location, so the relative path to ‘file\_1’ would be ‘documents/file\_1’.

# Basic commands

In this section I will touch upon some basic commands. In the directory with this document should be a file titled ‘linux-cheat-sheat.png’ which will contain more commands with explanations.

echo “<text>” – print <text> in the terminal

touch <name\_new\_file> - creates an empty file named as variable <name\_new\_file>

mv <path\_to\_file> <path\_to\_new\_location> - moves file to new location. This command can be also used to rename files, just move it to the same folder with new name.   
Example (using direcotry structure from previous section) – ‘mv /usr/local/documents/file\_1 /usr/local/file\_2’ would move ‘file\_1’ to ‘local’ directory and rename it to ‘file\_2’

vim <path\_to\_file> - opens file in a terminal text editor called vim. It is a useful tool for quickly modifying the files, but a bit hard to use. You also use commands to edit text in vim. There should be a file called ‘vim\_cheat\_sheat.jpg’ in the directory with this document – it explains basic vim commands.

rm <path\_to\_file> – commands to delete files. You can delete standard files with it, but if you want to delete a directory you need to add ‘-r’ flag to it.   
Example: ‘rm -r /usr/local/documents’ would delete directory ‘documents’ and all its contents.

chmod <modification\_type> <path\_to\_file> – modification of file’s permissions. Every file has three types of permissions: ‘read’, ‘write’ and ‘execute’. These permissions can be set as true or false. If ‘read’ is set to true you will be able to see the contents of the file. If ‘write’ is set to true you will be able to change the contents of the file. If ‘execute’ is set to true you will be able to run the file as a program.  
Example: ‘chmod +x /usr/local/documents/file\_1‘ will set ‘execute’ permission of the ‘file\_1’ as true.  
There is a little bit more to permissions (for example different permissions for different users) but we will cover that later.

When you are typing linux commands your best friend is the ‘tab’ key on your keyboard. It automatically fills your command when there is only one option to fill it. If there are more options, you can press ‘tab’ twice to list them all.

You can save results of a command to a file with ‘>’ or ‘>>’.   
‘>’ creates a new file and puts result of command in it.   
‘>>’ appends files if one exists or creates new one if it does not, and then puts result of the command in this file.  
Example: “ls > result.log” will put result of ls command (content of the folder) into file called ‘result.log’.

You can use linux commands to create programs performing different functions. Before you execute program like that it needs to have ‘execute’ permission. Then you can use it by typing path to it as a first thing in the command.   
Example: command ‘/usr/local/documents/file\_1’ or (if you are in ‘documents’ directory) ‘./file\_1’ will execute file called ‘fiel\_1’. All the lines in ‘file\_1’ will be executed as commands.

Example program ‘example\_script1.sh’ will be provided in the ‘examples’ directory of this repository.

## Exercises:

1. Write a program which will create file ‘cat.log’ containing line ‘I like my cat’ and then it will print content of ‘cat’log’ on the screen.
2. Research on your own how to use ‘grep’ command and ‘|’ (pipe) tool in linux commands. Then modify file ‘example\_script1.sh’ form ‘examples’ directory so when it prints contents of a directory it will only print files created during this script execution.

# Installing new programs

# Regexes