Essentials of Linux

Hans de Nivelle

February 27, 2014

Why do I insist on Linux in this class?

- Windows is designed for non-experts. It is designed in such a way that the computer makes all decions and the user has no control of his own computer.
- It is very easy to for a program to install another programm.
- In general, there is no notion of a program running on the computer. Programs always try to take over the complete computer.

  (By opening new windows, installing toolbars, controlling access to the file system.)

  Programs have their own file interfaces.
- There is no good method of file control in Windows. No good way of job control.
- Compilers under windows tend to be not standard compliant.

  There is no separation of editor/compiler/file control/running environment.

1 Files

Computers store information in files. A file is a sequence of bytes that is stored under a name on some device (hard disk, CD, USB drive). A file can be as long as you wish, until the device on which you want to store it is full. A file length of 0 is also allowed, although not very useful. In linux, every file is part of a directory. (In Windows, this is called folder) A directory is just a list of files. Every directory, except for the top level directory, is part of another directory. The files and the directories form a tree structure. The highest directory is called `/` The leaf files contain the data (sequences of bytes) All other files are directories.

A complete file name has the following form:

```
/dir1/dir2/dir3/.../dirn/name
```

The `/` symbol separates the different directories of the filename. The sequence `/dir1/dir2/dir3/.../dirn/` indicates the directory in which the file is stored, starting at the top level and moving down to the file. Examples of real file names are:

`/home/nivelle/teaching/ansic2008/linux.tex`

and

`/home/nivelle/conferences/ijcar2006/final/generation.pdf`

The first file is called `linux.tex`. It situated in directory `ansic2008`. The directory `ansic2008` is a subdirectory of `teaching`, which in turn is a subdirectory of `nivelle`. The directory `nivelle` is a subdirectory of `home`. The directory `home` is a subdirectory of `/`, which is the highest directory. All files in the file system can be reached from `/`.

It is possible to store files with the same name in different subdirectories. Such files are completely unrelated. For example, it is possible to have files `/home/nivelle/teaching/ansic2008/linux.tex` and
`/home/nivelle/papers/linux.tex` at the same time.
The directory structure helps you to order your files, and you should think about how you organize your directory structure. If you are following different lectures, you could store files that belong to different lectures in different subdirectories. If you create a file for the first time, you can choose its name by yourself, and you should choose it in such a way that you can see what is in the file from its name.

2 The Shell

In linux, the main way of communicating with the computer is by typing commands in a shell.

Form the shell, you start a program, the program does its work, and after that, it goes away.

A command has form `filename par1 par2 ...` The shell looks up the contents (a sequence of bytes) of filename, loads them into main memory and executes them. What you need to understand is: (nearly) Every command is just a filename. When you type the command, the contents of the file is loaded into memory and executed. For example, the file `/bin/ls` is a program that lists all files in the current directory. (Try it out) The file `/usr/games/banner` is a program that prints its argument big. Try `/usr/games/banner -w 40 hello` [1]

The part that follows after the file name (`par1 par2 ...`) are the parameters. They are passed to the program, and the program decides what it does with them. It can ignore them, try to open them as files, interpret them as numbers. (or print them big)

[1]If this example does not work, then first read the next paragraph about **which**

As seen in the previous section, file names can be pretty long. Fortunately, you almost never have to type a complete file name. If you type a command, the shell will look for this file in a list of subdirectories. This list of subdirectories is called *the path*. If you want to see it, type `echo $PATH` . Because `/bin` is in the list, you don't need to type. `/bin/ls`. It is sufficient to type `ls`. If you want to know where `ls` is found, type `which ls`.

3 Types of Files

You probably agree that there are different types of files. For example, there are mp3s, jpgs and text documents. But how does the computer know? The bytes in the files are the same.

In linux, the answer to this question is simple: The computer does not know. If you type a command, the computer will load the file and try to execute it. If the file was not supposed to be an executable, this probably results in some kind of error message.

If you type a filename as parameter to another program, like in `cat aa`, (which prints a file to the screen) it is the program that decides what it will do with the file name. If the file is not of the right type, the program may or may not notice this. For example, if you type `cat aa`, and aa is not a text file, some mess will be printed on the screen.

4 Running a C^{++}-program

1. Log in, start a shell.
2. Create a directory, if you didn't do this yet. Type `mkdir task1` (or another name)
3. Make the directory ansic your working directory. Type `cd task1`
4. Check which files you have in this directory. (If you just created it, there are no files) Type `ls` , or `ls -l` , or `ls -tl` .
5. Choose a name for your C^{++}-program, for example `task1.cpp` Type `emacs task1.cpp` . There are different editors. You can also choose another one, for example `vi task1.c` .

   Type the C-program, until you think that it works, and the code lay out is good. Save the file and leave the editor.
6. Type `ls`, to see if the file is there.
7. Compile the file. This is done by typing `g++ task1.c -o task1` If the compiler does not complain, it creates a file task1 (not to be confused with task1.c) that can be executed.
8. Type `./task1` . Remember that every command is just a file name.

5 Some Standard Directory Names

- `.` : Name of current directory. You need to type `./task` if you want to execute a program from the current directory. (Because the current directory is not in the path)
- `/` : Name of top level directory.
- `..` . Name of the directory that is one level above the current directory.
- `~` . Name of your home directory. For me, it is `/home/nivelle` .

6 Some More Commands

- `rm name` . Remove a file. In linux, removing means really removing! There is no trash can.
- `mv name1 name2` . Rename file. If file name2 exists, it is deleted without notice. If you don't want that, type `mv name1 name2 -i` .
- `cp name1 name2` . Copy file. If name2 already exists, it is overwritten without notice. If you don't want that, use `cp name1 name2 -i` .
- `ls`. List files in current directory.
  `ls dir`. List files in directory name. You can add parameters -l: Show the lenghts, creation date.
  -t : Sort files by time, newest files first.
  -lt : Sort, and show lenghts and creating date.
- `rmdir dir` . Delete directory dir. The directory must be empty.
- `cat file` . Type file name on screen (so that you can see what is in it, if it is a text file.
- `more file` . Same as cat, but it stops after every page.
- `cd dir` Change current working directory to dir.
  `cd ..` Change current working directory one level up.
  `cd ~` Change current working directory to your home directory.