Contents

Chapter 1

Brief synopsis of Linux

Here you have some basic, useful Linux commands. This synopsis is not meant to be exhaustive or to always offer the best way to achive one goal. It is rather intended to help a beginner use a Linux system and get a quick start in running the lab applications. The synopsis is mainly concerned with the Fedora distribution (the one which is used in the lab). Unless otherwise specified, the commands are to be typed at a terminal prompt. The argument(s) required should be obvious once you look at the example provided and at its description.

1.1 How to begin and end a work session

In the login screen, you should enter: you username (aka login name or account name or simply user) and your password, as provided by the administrator. Henceforthh, the username will be `iia`.

```
login:  iia
password:
```

< −− Please continue typing even if no echo is shown!

Typically, your current directory will be `/home/iia` (we'll call this your *home directory*). If you open a terminal, you will "arrive" in that directory. The home directory is denoted by ~ (the tilde character). Most of the time it is here that you find a file which has been copied from a remote computer (e.g., a file copied to you machine by your teacher), unless otherwise specified by the person who did this (and most of the time your teacher will not bother to specify a different destination).

To quit your session, search for `System -> Logout` in the upper right corner of the graphical interface.

1.2 Shut down a system: `poweroff`

Alternatively to typing `poweroff` at the prompt, you may use `System -> Shut down`, also available from the upper right corner of the graphical interface.

Pay attention! Before shutting down a computer, always do:

```
who
```

in a terminal, in order to see whether some other users are connected to your computer (don't forget Linux is actually a multiuser operating system – more users can work on the same machine in the same time and share its resources. See more details in Section 1.5).

1.3 Help for a command: `man`

Example:

```
man ls
```

shows a help for the command `ls` (list all files in a folder/directory).

1.4 Get the IP address of your computer: `ifconfig`

Your LAN interface is typically called `eth0`. The IP address consists of 4 dot-separated numbers, e.g., `192.168.1.3`

1.5 Connecting to another computer via secure shell: `ssh`

Example: If you are working on computer `c1` and want to connect to computer `c2`, in the `iia` account, you should do on `c1`:

```
ssh -X iia@c2
```

then type the password of `iia` on computer `c2` (even if you see no echo when typing). Here, `c2` is either the name or the IP address of the remote computer. You will get acces to a terminal on `c2` and be able to run programs there, including GUI featuring programs (due to the `X` option).

To close (i.e. end) the connection, just type in the remote terminal:

```
exit
```

1.6 Copy files on some other computer via secure copy: `scp`

Example: if you want to copy file `f1` from computer `c1` to destination computer `c2`, into account `iia`, open a terminal on `c1`, change directory (see Section 1.7.1) to the one containing `f1` and type:

`scp f1 iia@c2:` <−− Please note the colon at the end!

then type the password of `iia` on computer `c2`. `f1` is copied on computer `c2`, in `/home/iia`.

1.7 Directories and files

The directory system has a unique root called `/` (similar to `C:` in Windows). File names are **Case Sensitive**. Files have no specific extensions; a dot "`.`" might be a part of the file name (a file could be named `f`, `f.c` or even `f.1.2.3.c`).

1.7.1 Change current directory: `cd`

Example: change directory to `/home/iia/john`:

```
cd /home/iia/john
```

Example: change directory to the parent of the current directory:

```
 cd ..
```

Example: change directory to your home directory:

```
cd ~
```

or simply

```
cd
```

1.7.2 Print working directory (the name of the current directory): pwd

The command `pwd` prints the complete name of the current directory. Alternatively, you can do `echo $PWD` (see also Section 1.8).

1.7.3 Paths to files

Let us assume that the current directory is `/home/iia` and we create here a folder called `d1`, then, inside `d1`, a file called `f1.txt`. We can reffer to the file `f1.txt` either as `d1/f1.txt` (we will say we use the "relative path") or as `/home/iia/d1/f1.txt` (in this case, we use the "absolute path"). If we type `./d1/f1.txt`, we also use a relative path as a dot means "the current directory", which is, for now, `/home/iia` .

If we now change directory to `d1`, we can reffer to `f1.txt` as simply `f1.txt` or as `./f1.txt` (as you have probably noticed, the current directory is now `/home/iia/d1`), or even as `/home/iia/d1/f1.txt`. Please note the absolute path remains the same, while the relative path (not surprisingly) changes.

Here, by "path" we mean the name of a file preceded by the names of the directories containing it.

1.7.4 Symbolic Links: ln -s

This command creates a symbolic link (like a Windows shortcut: an alternative name for a file). It's usually used in order to give a "constant" name to a file whose "true" name changes over time (e.g., a library, whose name includes the version number). Example:

```
 ln -s /usr/local/lib/myjar.0.1.23.jar myjar.jar
```

1.7.5 Archives: tar (or via GUI)

To create a compressed archive called `archive_name.tar.gz`, you should do:

```
 tar zcvf archive_name.tar.gz f1 f2 d1
```

where `f1, f2, d1` are either files or directories to be added to the archive. When adding a directory to an archive its whole structure is preserved.

You can do the same job via a graphical interface, by right clicking on the file.

To decompress and extract files from an archive, you may use:

```
 tar zxvf archive_name.tar.gz
```

1.7.6 Midnight Commander (a Total Commander-like tool): `mc`

The `mc` command starts a tool which allows you to create, copy, delete files or directories by using the Functional Keys. One can see their functionalities at the bottom of the panels (e.g., `F7` allows creating a new directory).

1.8 Environment variables

They contain data which is important for the whole system. Examples:

- `PATH` contains the name of directories, separated by colons, in which a file launched into execution is to be searched
- `CLASSPATH` tells Java applications and JDK tools where to search for classes
- `PWD` gives the present working directory

The `env` command shows all environment variables and their values.

1.8.1 Setting: use `export`

You need to type:

```
VARIABLE=value
export VARIABLE
```

or:

```
export VARIABLE=value
```

Example:

```
export PATH=$PATH:/home/iia/d1
```

says the new value of `PATH` will be its old value (`$PATH`), to which we append (via `:`) the value `/home/iia/d1`

Example:

```
export PATH=$PATH:.
```

appends the value `.` (i.e., the ”current directory”) to `PATH` ; as you remember, one could always reffer to the current directory by using the character `.` (a dot).

Let us assume now the current directory is `/home/iia` and we create here a folder called `d2` and a file called `f1.sh` in `d2`, then we set the execution rights to `f1.sh` as explained in Section 1.10.3. In order to run `f1.sh`, we have the following options:

1. using the absolute path: `/home/iia/d2/f1.sh`
2. using the relative path: `./d2/f1.sh` or simply `d2/f1.sh`
3. if the directory containing `f1.sh` (i.e., `/home/iia/d2`) has been added to `PATH`, you can just type `f1.sh` and this will work no matter which your current directory is

4. change directory to d2 and type ./f1.sh

5. provided you have changed directory to d2, you can do a f1.sh but this only works if the current directory, denoted in Linux by a dot, has been added to the PATH. If this is not the case and the directory containing f1.sh has not been added to the PATH either, then simply typing f1.sh WILL NOT WORK and will return a command not found error.

Let us assume the dot is in the PATH, but no other directories have been added there, and you have a file called f1.sh in directory d1 and a file also called f1.sh in directory d2. If you type just f1.sh, you will be able to run f1.sh, but the version to be run depends on your current directory. So, if you do cd /home/iia/d1 then f1.sh, you will actually run /home/iia/d1/f1.sh. But if you do cd /home/iia/d2 then f1.sh, you will actually run /home/iia/d2/f1.sh.

The command export PATH=$PATH:. in the example above basicly tells that, if the user tries to run file x by typing x at the prompt, the shell must search x in the directory where the command x has been issued, besides the other directories in the PATH.

To find out the absolute path to a program progr which is run, type:

```
which progr
```

1.8.2 Displaying: echo $VARIABLE_NAME

If we want to see which is the current value of a specific environment variable, we may use:

```
echo $VARIABLE_NAME
```

Example:

```
echo $PATH
```

1.8.3 Automatic Setting

If you want directory /home/iia/d1 to be added to your PATH automatically each time you log in, please make sure the following lines are added to the file /home/iia/.bash_profile:

```
PATH=$PATH:/home/iia/d1

export PATH
```

(alternatively, you can use the file .bashrc in your home directory).

1.9 USB memory stick

Usually, when the stick is introduced into the computer, it gets automatically mounted on the file system. This means it temporarily becomes integrated into and visible as a part of the file system. From a terminal, you may access it in /run/media/iia/THE_NAME_OF_YOUR_STICK/. If a writting operation has been conducted over a file on the stick (like writting, modifying or deleting a file), then, before removing the stick from the computer, you must unmount (eject) it. A right click on the stick's icon will guide you.

1.10 Rights over files

1.10.1 Rights and types of users

In Linux, users of a file are divided into three categories: file owner, denoted by `u`; file owner group `g`; others `o`. The rights a user may have over a file are `r`, `w` and `x` meaning the right to read, write (i.e., modify) and execute (i.e., run) the file. A file that could be executed is either a binary file (for example, the result of compiling and linking a C source), or a text file, aka script, containing operating system commands (e.g., the PATH setting command above).

As an example: if you want to run a binary file or a script, you need to make sure its execution right is set. Otherwise you will get an error message (e.g., `Permission denied`), signaling that you have no permission to run that file.

1.10.2 Displaying rights: `ls -l`

Example:

```
ls -l f1
```

returns:

```
-rwxr-xr-- 1 iia students 32212 Oct 22 13:46 f1
```

The first column of the result contains 9 letters showing the rights over file `f1` of its owner (`rwx`) – so `iia` can read, write and execute the file, group members (`r-x`) – the members of the `students` group can read and execute the file, but can't modify it; the other users (`r--`) can only read it.

1.10.3 Changing rights: `chmod`

Example:

```
chmod ug+r file1
```

grants the owner (user `u`) and her group (`g`) the right ot read (`r`) the file `file1` (if the user who launches this command has the right to grant them).

Revoking the right of writting the file for everyone (user, group members and the others)is done by either of the following two commands:

```
chmod a-w file1
```

```
chmod -R a-w dir1
```

The latter recursively changes the rights over the files and directories in `dir1`, at every nesting level.

1.10.4 Changing file owner and group: `chown` and `chgrp`

Example: to make `iia` the new owner of `file1`, do:

```
chown iia file1
```

Example: to change group for `file1` into `students`, do:

```
chgrp students file1
```

1.11 Seek for a file: `locate` or `find`

Example: `locate passwd`

returns the full path to all files in the system whose name includes the string ”passwd”. It actually looks in a system database which stores data about files, so it might not produce up-to-date results, depending on the moment the database update has been done.

Example: `find /home -name "passwd" -print`

searches recursively in the file system, starting from `/home`, the files called `passwd`.

1.12 Search for a string inside a file: `grep`

Example:

```
grep abc file1
```

displays every line in `file1` containing the string `abc`. The string could actually be a regular expression (`grep` actually stands for ”get regular expression pattern”), e.g., `f*.txt` which means ”all strings starting with `f` and ending in `.txt`”.

1.13 Editing files: `mcedit`, `gedit`, `kile`

```
mcedit file_name
```

(or `F4` on the file in Midnight Commander). `mcedit` heavily relies on Function keys. One can see their functionalities at the bottom of the panels (e.g., `F7` does a search of a word in the file).

Some other options are:

```
gedit file_name
```

```
xemacs file_name
```

For productively writing texts using LaTeX, a very good Integrated Environment is `kile`.

1.14 Redirecting commands: `>`, `>>` and `<`

The result of a command can be sent into a file instead of being displyed on the screen. This is done by `> dest`, if you want the original `dest` file to be erased, or `>> dest` if you need it to be appended.

Example:

```
ls -l > all.txt
```

creates a detailed list with all files in the current directory which will be written in the file `all.txt`.

Similarly, one can instruct the system to read from a file `f.in` instead of the keyboard (to redirect the input). This is done by using `< f.in` (e.g., `go < f.in` makes `go` read from `f.in`).

1.15 Pipes: |

More programs could be pipelined such that the output of one is used as the input for the next; the operator used is `|`.

Example:

```
ls -l | grep *.c
```

produces a list of all files in a directory (`ls -l`); this list will be the input for `grep *.c`, which searches for all files whose name end in `.c`.

1.16 Compiling and running programs

1.16.1 C programs: gcc and make

For `.c` programs, just do:

```
gcc
```

Example: let us assume we have in the file `hw.c` the following C code:

```
#include<stdio.h>

void main() {
    printf("Hello, world!\n");
}
```

you may compile it from command line like this:

```
gcc hello.c -o hello.exe
```

This will produce the output `hello.exe`, which can be run by typing `./hello.exe`

For bigger projects, the following command usually helps:

```
make
```

It assumes there is a file called `Makefile` or `makefile` in the current directory; this file contains the commands to be actually launched for compiling the sources.

Quite often, projects could be compiled and installed using the following procedure:

```
./configure
make
make install
```

where the last line should be run with root priviledges.

1.16.2 Java programs: javac, java and ant

A Java program could be compiled using:

```
javac
```

Example: if we have the following code in the file `HelloWorld.java`:

```
public class HelloWorld {

 public static void main(String[] args) {
        System.out.println("Hello, World!");
 }

}
```

we can compile it with:

```
 javac HelloWorld.java
```

which will produce the file `HelloWorld.class`.

Running the program needs launching the Java Virtual Machine, which is done by:

```
 java
```

In order to run the example above, we can do:

```
 java HelloWorld
```

If all we have is a `jar` file, e.g., `hw.jar`, we can type:

```
 java -jar hw.jar
```

A program which does for a Java project pretty much the same job as `make` for C projects is `ant`.

1.17 Processes on the system

1.17.1 Listing processes and their status: ps

Example:

```
 ps -ef
```

could return something like:

```
...
iia       5712  2692  0 22:54 ?        00:00:00 kdeinit4: kio_file [kdein e
root      5713     2  0 22:55 ?        00:00:00 [kworker/2:2]
iia       5752  2481  0 22:57 pts/0    00:00:00 ps -ef
...
```

showing each program running on your system, together with its ”process identifier”, or pid, which is a unique number attached to each process, as seen in column 2.

1.17.2 Forcibly stopping a process: kill

```
 kill -9 pid
```

where `pid` is the unique number of the process you want to stop, as returned by `ps`.

1.18 Display a .pdf file from a terminal: `okular` or `evince`

Portable Document Format (.pdf):

```
okular file_name.pdf &
```

or

```
evince file_name.pdf&
```

1.19 Browsers: `firefox` or `google-chrome`

Mozilla Firefox, Google Chrome etc. could be launched via the GUI or from a command line, e.g., by typing `google-chrome&` in a terminal.

1.20 MS Office-like packages: `LibreOffice` or `Apache OpenOffice`

LibreOffice and Apache OpenOffice are two alternatives for MS Office. Corresponding to Word, Excel and PowerPoint, you have Writer, Calc and Impress.