Exercise 1 (1/3)

Motivation:

- Sometimes it is needed to create a separate File System in an exiting File System and you don't want to re-partition the disk to allocate another sub-partition for this new File System.
- There's a way to create a File System on a virtual device which (the device) can then be mounted to the original File System, providing you with a File System in a File System.

Exercise 1 (2/3)

Description:

- Create a file of a size you need. There're several ways.
- Setup a loopback device on the created file.
- Create a filesystem on the created file, which, in fact, already is a device.
- Mount the created filesystem on the existing (the one you're in) filesystem.

Hints:

- Some useful commands for this exercise: `dd`, `fallocate`, `mkfs`, `losetup`, `mount`.
- You need super user permissions to execute some commands.

Exercise 1 (3/3)

Constraints:

- You have to create a file *lofs.img* not less than 50 MB.
- You should set a loopback device on the created file.
- You have to create a filesystem for the loopback device.
- You have to create a directory `lofsdisk`.
- You have to mount the created filesystem on the mount point `lofsdisk`.
- You have to put all commands that you executed in a script *ex1.sh*.
- You have submitted the script *ex1.sh*.
- The results should be reproducible by running the script *ex1.sh*
- Don't forget to add `sudo` for commands which need that permission.

Exercise 2 (1/4)

Motivation:

- What if you want to run a process, that you don't trust. That you don't want to see your files and data? What if you could isolate it in such a way that the process will see only what you want it to see?
- There are multiple ways to achieve it. One of which is `NameSpaces`. But today we will use another older way - `chroot`. This command allows you to change what is the *root* dir for the process.
- For example, you could create a separate file system, mount it onto a virtual device (loopback) on a file, create a process and chroot it on this file. Everything that the process will then create can be transfered as a single file. Or removed. Or.. whatever.

Exercise 2 (2/4)

Description:

- Using the file from Exercise 1 (create it, loopback it, mount it, make sure, it is accessible)
- Put a few files in this virtual mounted file system.
- Create a simple process (write a C program) that lists the contents of the `root` (/) of the file system.
- Run this process and `chroot` it so that in the output of your process will only contain the files you created earlier.
- Run the process again without `chroot` and prove that now the process sees the actual `root`

Note: If you `chroot` and there are no binaries like `cd`, `touch`, etc., you may want to first copy them to the chrooted location.

Exercise 2 (3/4)

Constraints:

- You have finished Exercise 1 and mounted the created Loopback File System (LOFS). Otherwise, return to Exercise 1.
- Add two files file1, file2 to the LOFS where file1 contains your first name, and file2 contains your last name.
- Write a C program `ex2.c` which will list the contents of the `root` directory (/). Compile `ex2.c` to `ex2.out`.
- Add `bash, cat, echo, ls` commands to the LOFS. Add their shared libraries too, otherwise they won't work.
- Change the root directory of the process to the mount point of the created LOFS and run the program `ex2.out`. Save the output of the program in a file `ex2.txt`.

Exercise 2 (4/4)

Constraints: (cont.)

- Run the same program again (DON'T change the root directory of the process). Append the output to the file `ex2.txt`.
- You have to put all commands that you executed in a script *ex2.sh*.
- You have submitted all files *ex2.c*, *ex2.txt*, *ex1.sh*, *ex2.sh*.
- The results should be reproducible by running the scripts *ex1.sh*, *ex2.sh*
- Don't forget to add `sudo` for commands which need that permission.