Organisation

- Work in a team of 2 or 3 students
- The duration of this lab is 4 periods (2 weeks)

Pedagogical objectives

- Become familiar with the Unix permission model
- Verify and correct file permissions manually
- Convert the manual steps into a simple tool for users
- Become familiar with the `find` command

Task 1: Exercises

Interpreting account and group information

Execute the command `id` to display information about your account and the groups you are member of.

- What is your UID and what is your account name?
- What is the GID of your primary group ("groupe principal") and what is its name?
- How many other groups are you a member of?

Interpreting access control metadata on files and directories

1. For the following files, determine who is the owner, which group owns the file and characterize the group of people who can read, who can write and who can execute the file.
    - `/etc/passwd`
    - `/bin/ls`
    - `~/.bashrc`
    - `~/.bash_history`
2. Examine the permissions of your home directory (what option do you have to pass to `ls` to examine the permissions of directories?).
    - Who is the owner and which is the owning group?
    - What is the configuration of permissions?
        - Who can list files?
        - Who can create files?
3. What permissions allow you to create files in the `/tmp` directory?

Modifying access rights

1. Create a file and initialize its permissions to `rw- --- ---` with the following commands:

    ```
    touch file
    chmod 600 file
    ```

    Using `chmod` in symbolic mode, create the following configurations (from initial configuration "600"):

```
rw- r-- ---
rwx r-x ---
r-- r-- r--
rwx r-- r--
rwx --- ---
```

2. Conflicting permissions - Create a file (you are going to be the owner) where the permissions are configured to
   - not allow the owner or the group to write to the file,
   - but allow the other users to write to the file.

   What does the OS do if you try to write to this file?

Giving other users access to your files

Do this exercise with another student. Both of you log in to the shared server at `ads.iict.ch`.

1. Is your colleague able to read the files in your home directory? If yes, why? If no, why not?
2. What do you need to do so that your colleague (and maybe others) can read your files? What do you need to do so that nobody else can read your files?
3. In your home directory create a directory named `shared`. By using the commands `chmod` and `chgrp` configure the directory in such a way that **only** your colleague is able to read the directory and its files. You can use the groups `proj_a` and `proj_b`. Both you and your colleague are already a member of these groups. (For the sake of this exercise, suppose that nobody else is member of this group, only you and your colleague.) Give your colleague also access rights to create new files and modify existing files.

   What commands did you use?

Note: Most Linux distributions make the users' home directories open to everyone (read access). Ubuntu adopts a new policy in release 21.04 to make home directories private, this feature has been desactivated on our server.

Find

`find` is a powerful Unix command to search the files on a computer by name, by size, or other criteria.

1. Pro tip: If you run `find` without any search criteria, it will simply display everything it comes accross in its recursive traversal. So to list recursively everything in the current directory, type simply

   ```
   find .
   ```

   What does find do with hidden files or directories?

2. Using `find` display all the files in your home directory
   - that end in `.c`, `.cpp` or in `.sh`

    - that are executable
    - that have not been modified since more than two years
    - that have not been accessed since more than two years
    - that have not been accessed since more than three years and that are bigger than 3 MB (good candidates for cleanup)

  Display all the directories in your home directory

    - that are called `.git` (probably the root of a git repository)

3. Suppose your current directory has no subdirectories. You want to display all files that contain the word `root`. Which of the two commands is correct and why:

```
find . -type f -exec grep -l 'root' {} \;
find * -type f -exec grep -l 'root' {} \;
```

Task 2: Display world-writable files

Context

While working with their files users may accidentally set permissions that are too loose and create security problems. As a system administrator you want to provide your users with a tool that they can use to scan their files, detect problematic permissions and correct them.

Tool for security checks

The tool will allow users to detect files or directories that are "world-writable". World-writable means the write bit is set for "others". This is obviously a security risk.

Create a test directory that you will use while developing the tool. Create a directory named `test_dir` and in it create a few files and subdirectories. Change the permissions on some files and some directories using `chmod` so that the write bit is set for "others".

Use the `find` command on the test directory to list all files and directories. Then modify the command to find only the world-writable ones.

Write a script named `fix_permissions` that for the time being only displays the world-writable files and directories. The output should look as follows:

```
The following files/directories are world-writable:
test_dir/foo
test_dir/dir1
test_dir/dir2
```

Task 3: Pass the directory as an argument

The user should be able to specify which directory the tool should analyse. Modify the script such that it takes one argument, the directory. The script should behave as follows:

- If it is called without argument it should display a corresponding error message and exit with a return code 1.

- If the given argument is not a valid directory it should display a corresponding error message and exit with a return code 1.

Task 4: Propose a fix

The tool should not only diagnose problems, but also offer a fix, that is, remove the offending permissions from the files. But the tool should not do that automatically, it should first ask the user if he wants to do this.

Modify the script such that after displaying the world-writable files it asks the user the question

```
Do you want the permissions to be fixed (y/n)?
```

If the user responds affirmatively the script proceeds and removes the permissions.

Note: The tool should ask only once for all the files, not for each file individually.

Tip: To avoid having to re-create the initial state of the test directory every time you run the script do the following: Run the script on a copy of the test directory that you throw away after use.

Task 5: Display group-writable files

To prepare for this task, create two groups on your local machine by becoming the superuser:

```
sudo addgroup proj_a
sudo addgroup proj_b
```

Then add yourself to both groups (replace `albert.einstein` with your user name):

```
sudo adduser albert.einstein proj_a
sudo adduser albert.einstein proj_b
```

In most companies the employees work in teams and they need to share documents, source code or other files with their colleagues. To enable this sharing on the Unix file system level a user needs to change the permissions on files and directories.

A file or directory that is shared in a team

- is assigned a group that corresponds to the team (each team member becomes member of the group),
- has write permissions for the group.

Sometimes a user forgets that he shares certain files with others. It is helpful to remind the user which files and directories are shared with the teams. Extend the script to show all files and directories that are writable for the group. However do not include files or directories assigned to the personal group of the user (the personal group is the one that has the same name as the user name).

The script should produce output like the following:

```
The following files/directories are writable for groups:
test_dir/dir3
test_dir/baz
```

To test the script extend your test directory with files that belong to project groups and are group-writable.

Hints:

- On Ubuntu Linux the personal group of the user has the same name as the user (this is a standard practice on almost all modern Linux distributions).
- The name of the current user is available in the environment variable `$USER`.
- You do not need to verify which groups the user is a member of. When the file has a group owner that is not the personal group of the user there is a potential security problem, and the file should be displayed.

Lab deliverables

Deliver a report in PDF format that contains answers to questions, and if you are asked to write a script, the listing of the script.

For the test directory create a listing with the following command and include it in the report:

```
ls -lR test_dir > test_dir.txt
```

Create a zip archive in which you put:

- The report
- The script you wrote (use the name specified in the task).
- The listing of the test directory.