

How to set permissions within a shared project

Lukas Fischer
SAFE Research Datacenter*

Last Update: December 6, 2023

Within the SAFE research cluster, project permissions are managed through groups. When a new project is created, a corresponding group is automatically generated, comprising all users with access to the project on the cluster.

There are two primary methods for adding files to the project:

1. Mounted network drive under Windows (Guide): The permissions for folders in the project drive are automatically overridden, granting access to all users within the group.
2. Accessing the project drive via cluster mount: In this case, users define the permissions for folders/data.

This guide concentrates on accessing a project drive via cluster mount (2) and provides instructions on correctly configuring access rights to ensure all group members can access and work with files, not just the owner. It's crucial to emphasize that the owner of the files must set the rights accurately, as only this group member (and HRZ Admins) has the authority to modify them.

Contents

1	Contact	2
2	Explanation of permissions set on Linux	2
3	Setting folder and file permissions within your code using python.	3
4	Change folder and file rights for specific folder (and its subfolders) using python.	3
5	Change folder and file rights for specific folder (and its subfolders) using Windows PowerShell.	4

*fischer@safe-frankfurt.de

1 Contact

If you encounter any difficulties or just want general information, do not hesitate to contact us.

SAFE Research Datacenter: datacenter@safe-frankfurt.de

More information about the SAFE Research Datacenter, and further guides can be found [here](#), and [here](#).

2 Explanation of permissions set on Linux

In Linux, file and directory permissions dictate who can perform specific actions, and they are organized into three sets: one for the owner, another for the members of the file's group, and a final set for everyone else. These permissions regulate actions such as reading, modifying, or executing a file or directory. The `chmod` command is used to set these permissions, and to view them, we employ `ls`.

One approach to using `chmod` is providing the desired permissions for the owner, group, and others as a three-digit number. The leftmost digit represents the owner's permissions, the middle digit for group members, and the rightmost digit for others. The digits are determined using the following guide:

- Start at 0
- Add 4 for read permission
- Add 2 for write permission
- Add 1 for execute permission

The possible digits and their meanings are as follows:

- 0: (000) No permission.
- 1: (001) Execute permission.
- 2: (010) Write permission.
- 3: (011) Write and execute permissions.
- 4: (100) Read permission.
- 5: (101) Read and execute permissions.
- 6: (110) Read and write permissions.
- 7: (111) Read, write, and execute permissions.

In our projects, we generally want all group members to have full access (read, write, and execute) to folders, while others should have no permissions. Therefore, the owner and group members' permissions should be set to 7, and others should be set to 0, resulting in `chmod 770`. Additionally, `chmod` has advanced options constituting a fourth access level beyond user, group, and others. Special permissions offer additional privileges. The special permission `SGID` (`setgid`) is relevant to our case. When applied to a directory, it causes new files and subdirectories created within it to inherit its group ID. The `setgid` permission has a numerical value of 2. Consequently, the correct `chmod` code for folders within a shared project on the cluster is `2770`. Similarly, for files within a shared project on the cluster, the `chmod` code is `0660`, as there is no need to allow execution or grant any additional special permission.

The following sections demonstrate different methods for correctly setting these permissions.

3 Setting folder and file permissions within your code using python.

To set folder permissions to 2770 and file rights to 0660, define the permissions using `os.chmod`. It is essential to note that Python expects you to include a leading 0 and o in front of your permissions.

```
1      # Import packages
2      import os
3
4      # Create new folder
5      os.makedirs(folder_path, exist_ok=True)
6
7      # Set permissions for folder
8      os.chmod(folder_path, 0o2770)
9
10     # Create and save a text file in the folder
11     file_path = os.path.join(folder_path, "random_text.txt")
12     with open(file_path, 'w', encoding='utf-8') as file:
13         file.write(random_text)
14
15     # Set permissions for the file
16     os.chmod(file_path, 0o0660)
17
```

4 Change folder and file rights for specific folder (and its subfolders) using python.

If you've created multiple folders and files and forgot to set the permissions directly, you can utilize the following code to configure the rights for all folders and files automatically.

```
1      # Import packages
2      import os
3
4      # Adjust permissions
5      def set_permissions(path):
6          # Set permissions for the folder and its subfolders
7          os.chmod(path, 0o2770)
8          for root, dirs, files in os.walk(path):
9              for d in dirs:
10                 folder_path = root + '/' + d
11                 os.chmod(folder_path, 0o2770)
12
13             # Set permissions for the files
14             for f in files:
15                 file_path = root + '/' + f
16                 os.chmod(file_path, 0o0660)
17
18     # Set path of your folder and execute function
19     input_path = path_with_wrong_permissions
20     set_permissions(path = input_path)
21
```

5 Change folder and file rights for specific folder (and its subfolders) using Windows PowerShell.

Another option to adjust permissions for your folders and files is to modify them directly on the cluster, for instance, using Windows PowerShell. First, log onto the cluster via ssh and navigate to the directory of your project or the top folder whose permissions you want to alter:

```
1 $ cd /SAFE_NAS/project_name/  
2
```

Next, examine all folders that do not have permissions set to 2770 and adjust them accordingly:

```
1 $ find . -type d -not -perm 2770 -print0 | xargs -0 -r chmod 2770  
2
```

Finally, inspect all files for permissions other than 0660 and adjust them accordingly:

```
1 $ find . -type f -not -perm 0660 -print0 | xargs -0 -r chmod 0660  
2
```