# 5. Setting filesystem ACL

To create permissions to files on your Samba server, you should use *ACL* (*Access Control List*). *ACL* allows you to set permissions for groups or final users.

The standard commands for working with ACL, *setfacl* and *getfacl*, are well detailed in the manual, so we would rather stick to examples.

After configuring the Samba server, a path to the common mounting point will be created, /var/calculate/server-data/samba/share.

Note: Make sure the filesystem on the disk that stores this directory supports *ACL*. *Do not forget to specify _acl* as a mounting option (not needed for XFS)._

## Add a new resource

Suppose we had two groups, manager and logist, as well as two users, john and peter.

Let us create the Manager directory, which all users, belonging to the group manager, will be allowed to read and write:

```
mkdir -m 700 Manager
setfacl -m d:g:manager:rwx,g:manager:rwx Manager
```

The first command shown above will create a directory with read, write and execute rights for root. The second command sets the same permissions for all users that belong to the manager group.

Whenever the root user or any user of the manager group creates files and directories, they will will inherit the access attributes.

## View permissions

If you need to check access rights to a directory, you can either do it from the console or use the Dolphin file manager, if you are running Calculate Linux Desktop (KDE edition).

To view permissions in console mode, execute:

```
getfacl Manager
```

The program will return the following:

```
# file: \320\234\320\265\320\275\320\265\320\264\320\266\320\265\321\200/
# owner: root
# group: root
user::rwx
group::---
group:manager:rwx
mask::rwx
other::---
default:user::rwx
default:group::---
default:group:manager:rwx
default:mask::rwx
default:other::---
```

In Dolphin, right-click on the file/directory, select the Permissions tab, then press the *Advanced Permissions* button. Enlarge the window to see all the attributes.

## Manage permissions

Use *setfacl* to manage permissions.
To modify an existing rule or add a new one, use the -m parameter, as shown below.

```
-m user:[user_name]:permissions[,user:user_name:permissions]
-m group:[group_name]:permissions[,group:group_name:permissions]
```

If the *user* option is omitted, *permissions* will be set for the file owner.
If the *group* option is omitted, *permissions* will be set for the group that owns the file.

**Example**

Granting the users john and peter with the read/write access rights to the file secretinfo:

```
setfacl -m user:john:rw,u:peter:rw secretinfo
```

Granting the group manager with the read/execute access rights to the file runit:

```
setfacl -m group:manager:rx runit
```

Multiple permissions can be combined in one line.

**Example**

Granting the user john with the read right to the qwerty file, the user peter with the read/write rights to the same file, the group logist with the read/execute rights, while the workers group will have no access rights to the file whatsoever:

```
setfacl --modify u:john:r,u:peter:rw,g:workers:-,g:logist:rx qwerty
```

## Default permissions

For directories, you can set ACL rights that will be assigned by defaults to files and directories created inside it. To do so, use the default identificator or the -d parameter. However, the default permissions will not be applied to the first directory.

**Example**

Setting the read/write permissions to all files and directories created in managerdata, for the user john:

```
setfacl -m default:user:john:rw managerdata
```

## Set permissions recursively

To assign permissions to all files and directories in a given directory, use the -R parameter. Since r, the read access rights are needed to read a file, while you will need rx, the read and execute rights to read a directory, you should probably specify X instead of x when setting permissions. The X flag assigns permissions only to those files and directories that already have the execute access right. The X flag is calculated on the moment when setfacl is launched, that is why it is interpreted as x in the default rules.

**Example**

Allowing the user peter to read all files and directories in folder:

```
setfacl -R -m d:u:peter:rwx,u:peter:rwX folder
```

or

```
setfacl -R -m d:u:peter:rw,u:peter:rwX folder
```

Both commands have a limitation that manifests itself when you create new files or directories. While in the first case, the peter user will be granted the execute right to all the newly created files in folder and its subdirectories, in the second case the user will not be able to read the new directories.

## Remove permissions

To remove an access rule for a user or a group, use the -x parameter.

**Example**

Removing access to the secretfile file for the user john:

```
setfacl -x u:john secretfile
```

Removing access to all files and directories in folder for the user peter:

```
setfacl -R -x u:peter folder
```

## Clear the access rules

To clear ACL rules, use -b. This parameter can be combined with -m, if you want to substitute permissions.

**Example**

Clearing all ACL rules and allow the john user to read secretfile:

```
setfacl -b -m u:john:r secretfile
```

## Copy ACL

If you need to assign *ACL* to a file analagous to some other file, use the following command:

```
getfacl basefile | setfacl -b -M - targetfile
```

targetfile will thus have the same permissions as basefile. The ~~M~~ parameter does that; it allows you to set permissions from a file (or a standard input stream, ).

## Effective mask

Effective masking is used to restrict action(s) for all users and groups, that are defined in *ACL*. It means, for example, that you can prevent all users from writing to a file by setting the effective mask r-x.

**Example**

```
setfacl -m m::rx filename
```

Sometimes, the effective mask is calculated automatically (for instance, when you run chmod, as well as when you create a file, because the umask permission is set). To initialize a mask, set it to rwx.

## Permission precedence rules

Whether a user can work with a file or a directory is determined by the priority rules described below:

1. If the user is the file owner, the owner's permissions are applied.
2. User-specific rights to this file are applied.
3. Access is granted, if the user belongs to at least one group that has access to it.
4. If the user does not belong to any group defined in *ACL*, the other permissions are applied.