ACCESS CONTROL LIST (ACL)

WestGrid Webinar. October 30th , 2019

Sergiy Stepanenko

Introduction into Access Control List

What is a purpose of ACL and why is it important?

Access Control is a measure of **authorization** and it answers the basic question "***Does subject S has right R for object O?***“

Subject **Right** **Object**

Fundamentals of **computer system security** are:

1. Authentication
2. Authorization
3. Audit

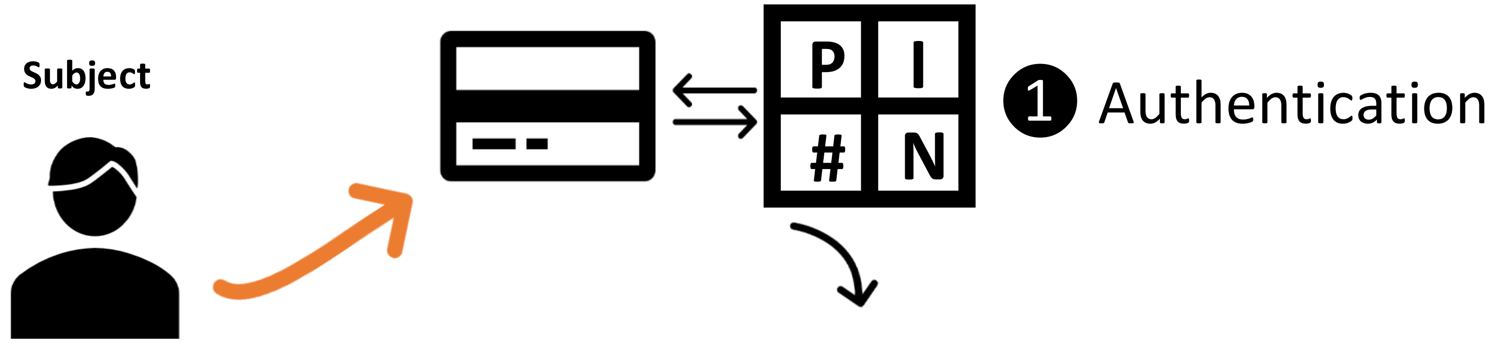
Introduction into Access Control List

Computer system security is very much like **financial security**, where a **person**:

authenticated by a combination of a card and PIN;

authorized by account type and balance;

audited by bank transaction records.



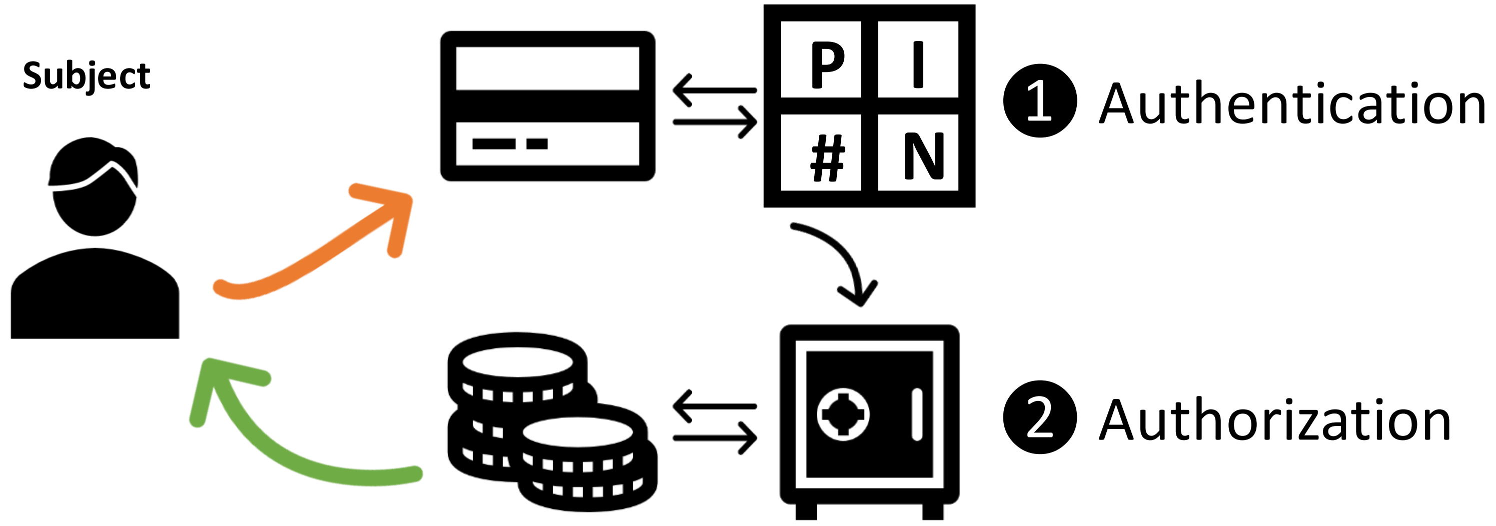
Introduction into Access Control List

Computer system security is very much like **financial security**, where a **person**:

authenticated by a combination of a card and PIN;

authorized by account type and balance;

audited by bank transaction records.



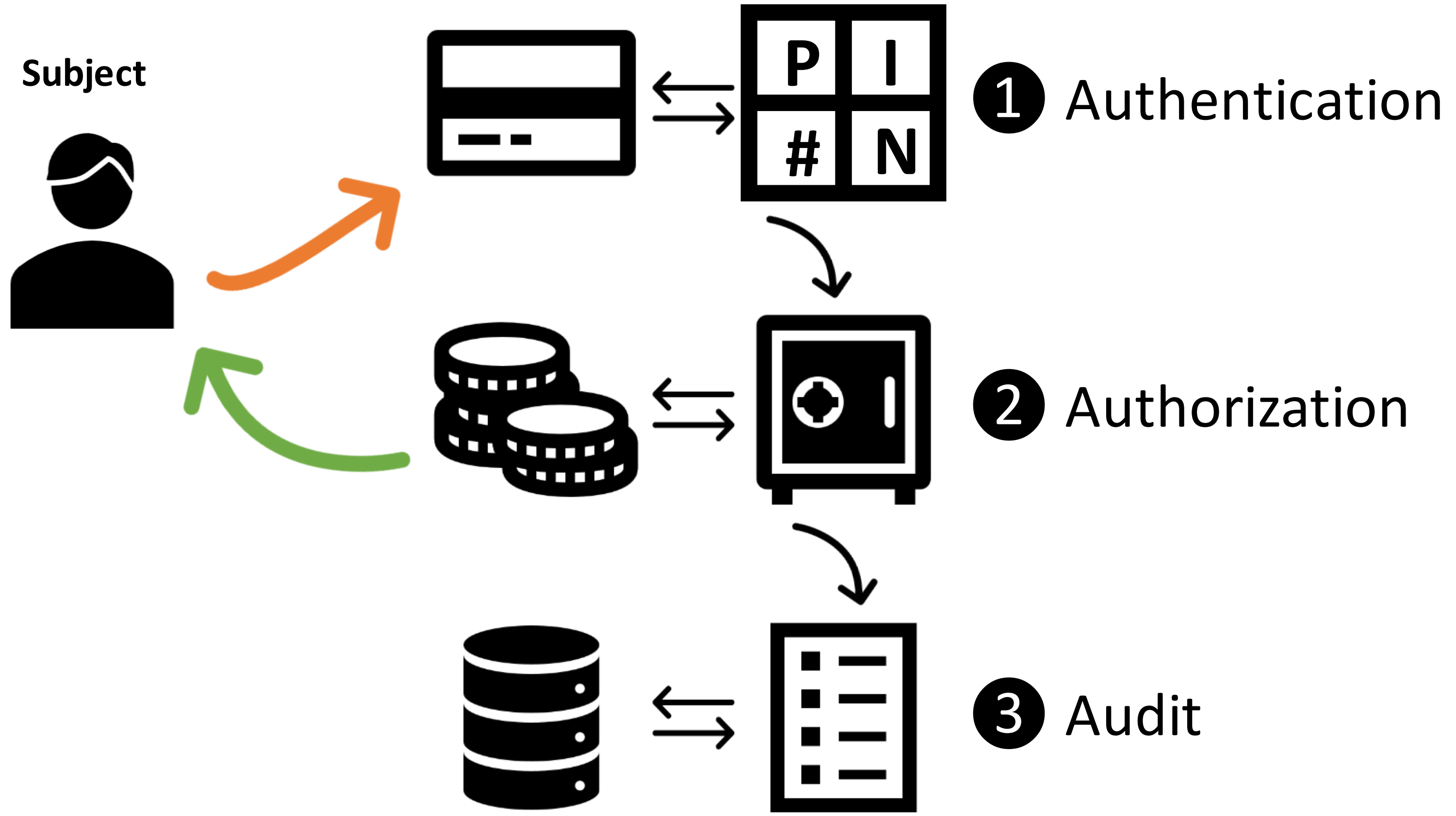
Introduction into Access Control List

Computer system security is very much like **financial security**, where a **person**:

authenticated by a combination of a card and PIN;

authorized by account type and balance;

audited by bank transaction records.



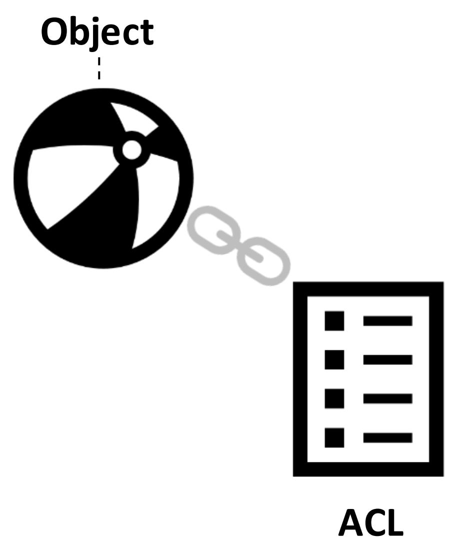
Introduction into Access Control List

There are many and, often, different implementations:

File System, **Network**, **SQL**, **Protocol** etc.

Common features of all ACLs are:

- Definition of a **subject**: user, group and other implementation-specific types
- Definition of **access type**: read, write, execute, list and other implementation-specific types
- Definition of **object**: file, directory, IP address, port, data-set and other implementation-specific types



Introduction into Access Control List

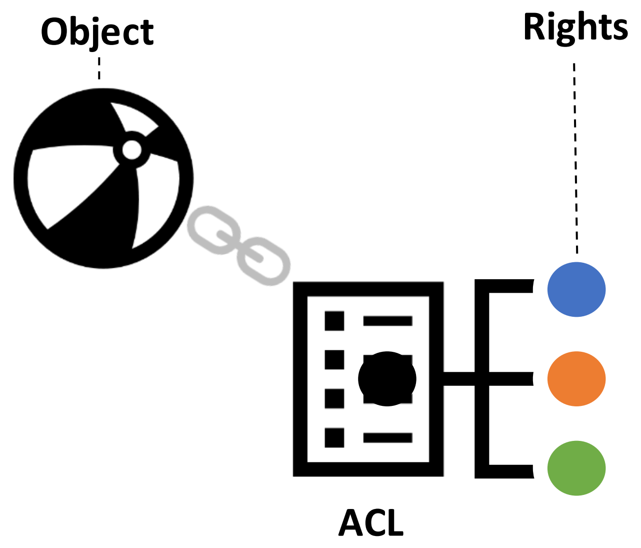
There are many and, often, different implementations:

File System, **Network**, **SQL**, **Protocol** etc.

Very important detail of ACL, as implementation of Access Control, is that ACL is **attached to the Object it controls**, not to the Subject, having certain rights to the Object.

Common features of all ACLs are:

- Definition of a **subject**: user, group and other implementation-specific types
- Definition of **access type**: read, write, execute, list and other implementation-specific types
- Definition of **object**: file, directory, IP address, port, data-set and other implementation-specific types



Introduction into Access Control List

There are many and, often, different implementations:

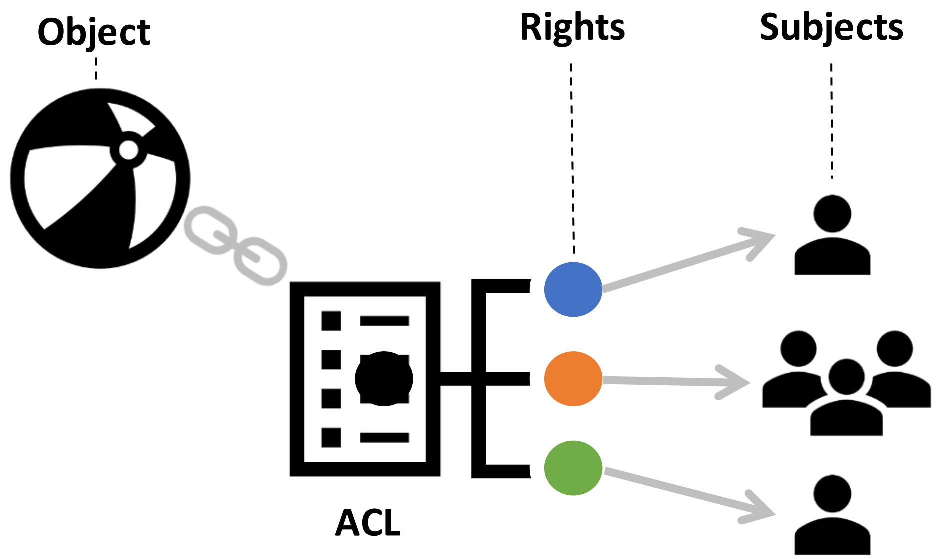
File System, **Network**, **SQL**, **Protocol** etc.

Very <u>important</u> detail of ACL, as implementation of Access Control, is that ACL is **attached to the Object it controls**, not to the Subject, having certain rights to the Object.

Implication of such implementation – it is trivial to find a **set all subjects** with any right to a specific **object**, but it is extremely difficult (although possible) to find a **set of objects** a specific **subject** has any right to.

Common features of all ACLs are:

- Definition of a **subject**: user, group and other implementation-specific types
- Definition of **access type**: read, write, execute, list and other implementation-specific types
- Definition of **object**: file, directory, IP address, port, data-set and other implementation-specific types



Implementation of ACL in POSIX-compliant file systems

There are hundreds of file systems, classified by their purpose, implementation and **supported standards**.

Some of them have an implementation of ACL, others – do not. I will not be able to cover all file systems and will concentrate only on those that are **POSIX**-compliant and used, primarily, on **Linux/Unix-based** systems.

These are: **Lustre**, **GPFS** (Spectrum Scale), **EXT** (2,3,4) and some others. They all have POSIX-type Access Control policies and, therefore, represent very **compatible** ACL implementations.

Implementation of ACL in POSIX-compliant file systems

Objects of ACL in file system are either **File** or **Directory**, Rights are: **Read**, **Write**, **List/Execute** and Subjects are: **User**, **Group** or **World** (Everyone).

File

Directory

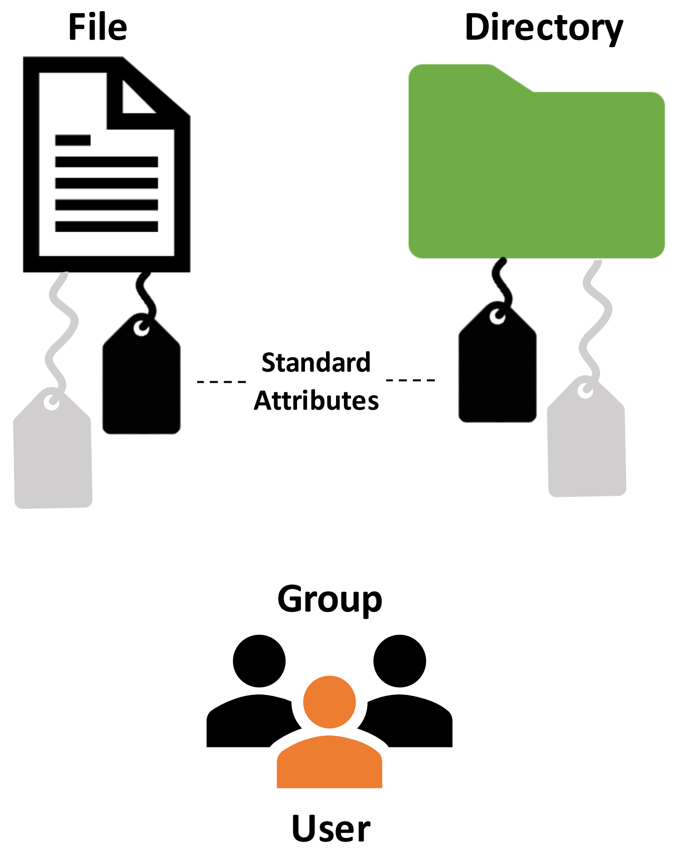
Group

User

Implementation of ACL in POSIX-compliant file systems

Objects of ACL in file system are either **File** or **Directory**, Rights are: **Read**, **Write**, **List/Execute** and Subjects are: **User**, **Group** or **World** (Everyone).

Objects – file or directory, have sets of attributes: **Standard** and **Extended**.

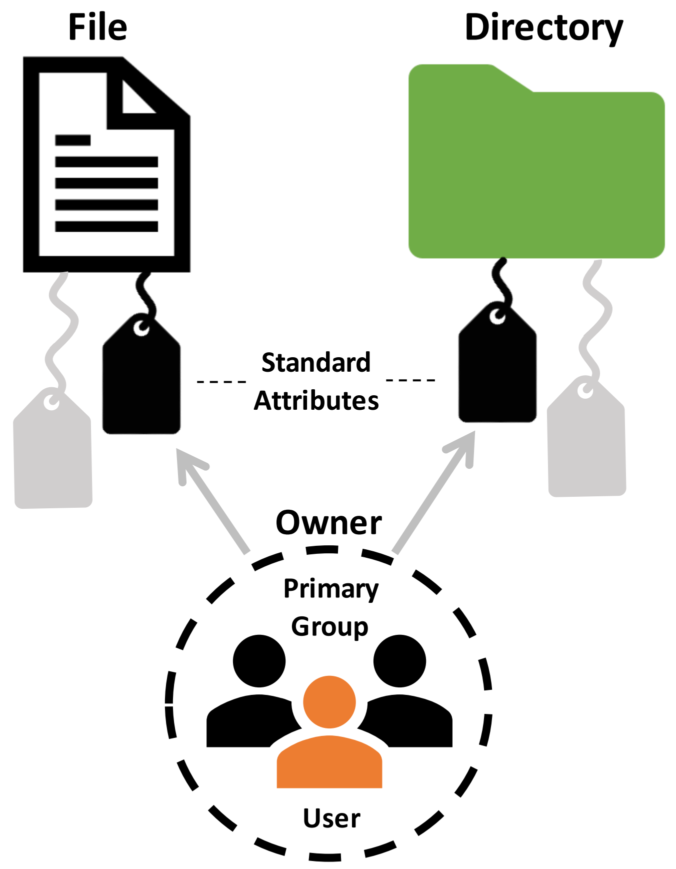


Implementation of ACL in POSIX-compliant file systems

Objects of ACL in file system are either **File** or **Directory**, Rights are: **Read**, **Write**, **List/Execute** and Subjects are: **User**, **Group** or **World** (Everyone).

Objects – file or directory, have sets of attributes: **Standard** and **Extended**.

By **default**, without ACL attached to a file or a directory, standard attributes determine access control: **Owner has Rights to Object**.



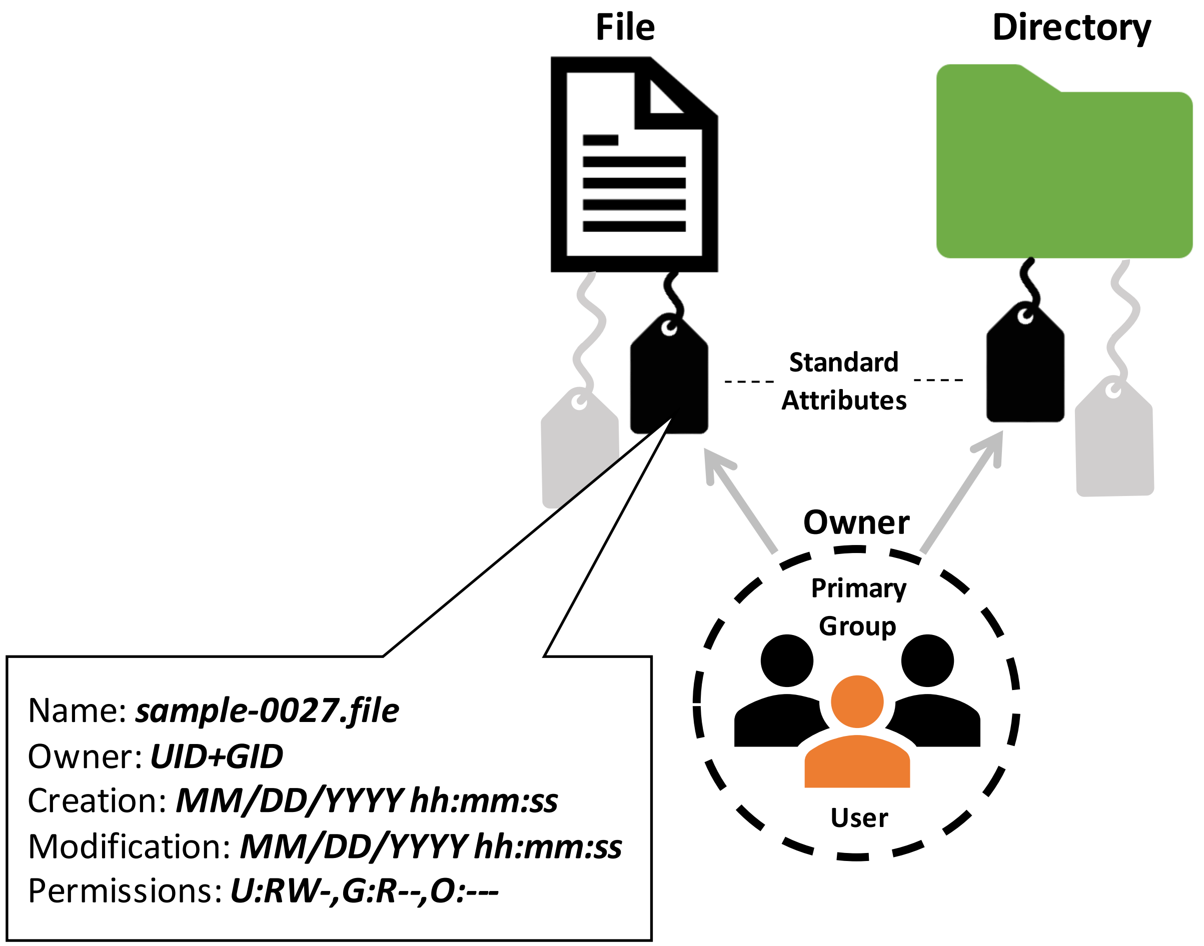
Implementation of ACL in POSIX-compliant file systems

Objects of ACL in file system are either **File** or **Directory**, Rights are: **Read**, **Write**, **List/Execute** and Subjects are: **User**, **Group** or **World** (Everyone).

Objects – file or directory, have sets of attributes: **Standard** and **Extended**.

By **default**, without ACL attached to a file or a directory, standard attributes determine access control: **Owner has Rights to Object**.

Since owner of a file or directory, typically, represented by **UID** (User ID) and **GID** (Group ID) of its creator – **User** and its **Primary Group** have certain rights to file or directory.

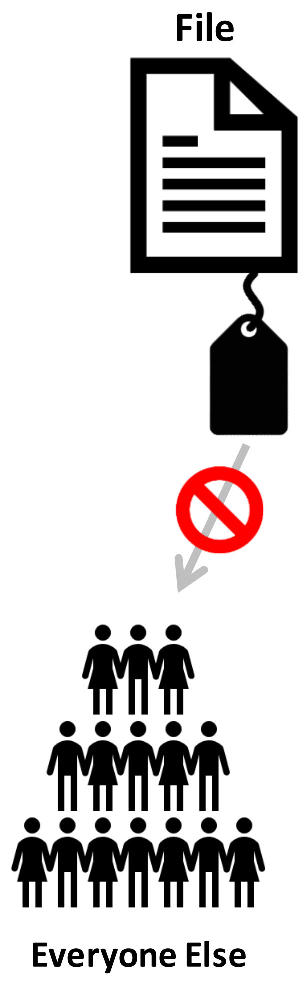


Name: ***sample-0027.file***
Owner: ***UID+GID***
Creation: ***MM/DD/YYYY hh:mm:ss***
Modification: ***MM/DD/YYYY hh:mm:ss***
Permissions: ***U:RW-,G:R--,O:---***

Implementation of ACL in POSIX-compliant file systems

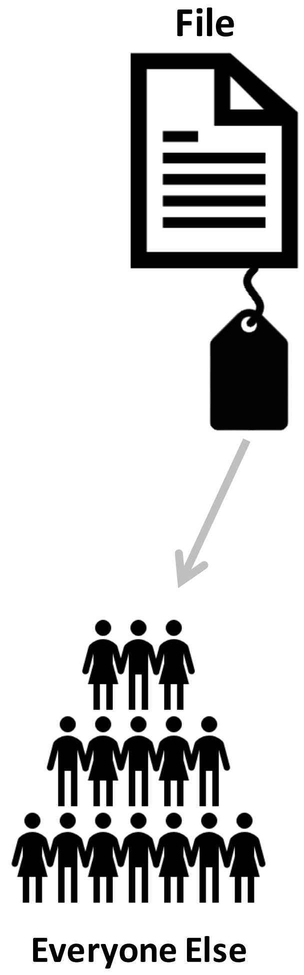
Additionally, Operating System may enable or disable access rights to a file or directory for **Everyone** (besides owner).

This is based on Operating System default security settings, which dictate Access Control levels for file system. Often, **Everyone** are not allowed any access.



Implementation of ACL in POSIX-compliant file systems

Additionally, Operating System may enable or disable access rights to a file or directory for **Everyone** (besides owner).



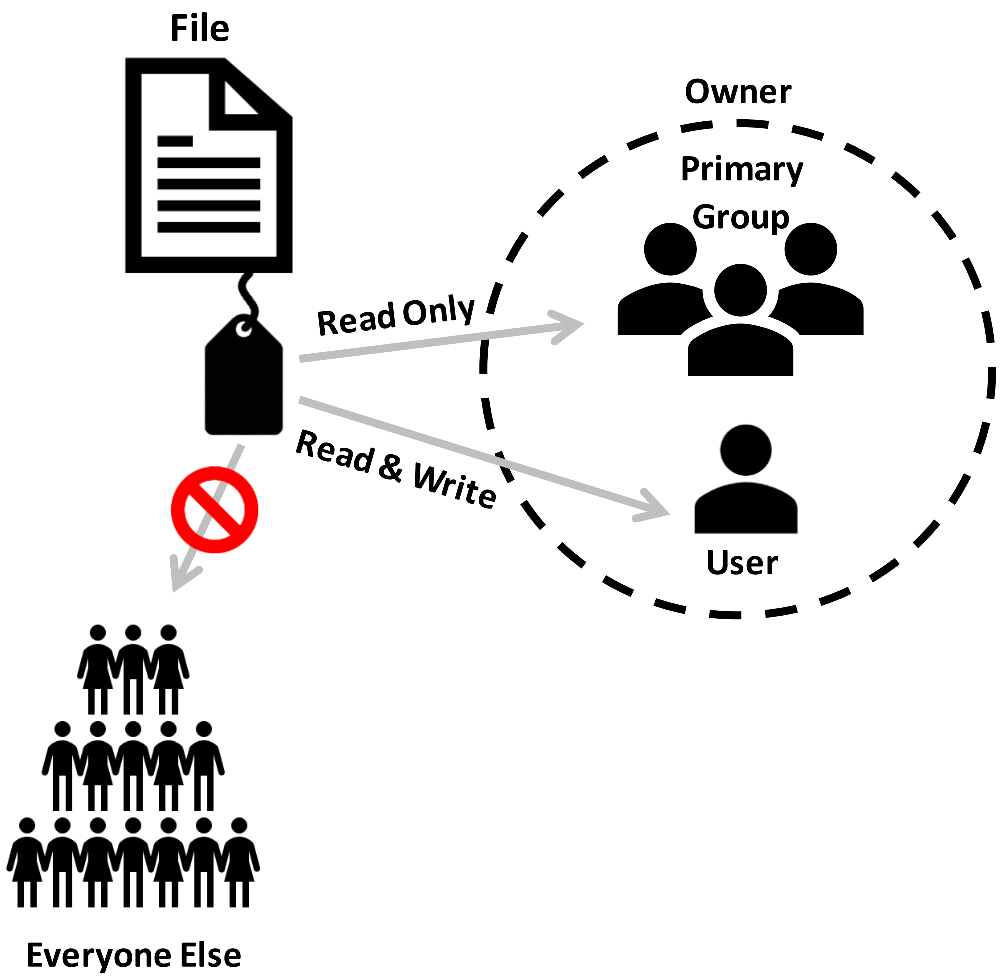
Implementation of ACL in POSIX-compliant file systems

Additionally, Operating System may enable or disable access rights to a file or directory for **Everyone** (besides owner).

This is based on Operating System default security settings, which dictate Access Control levels for file system. Often, **Everyone** are not allowed any access.

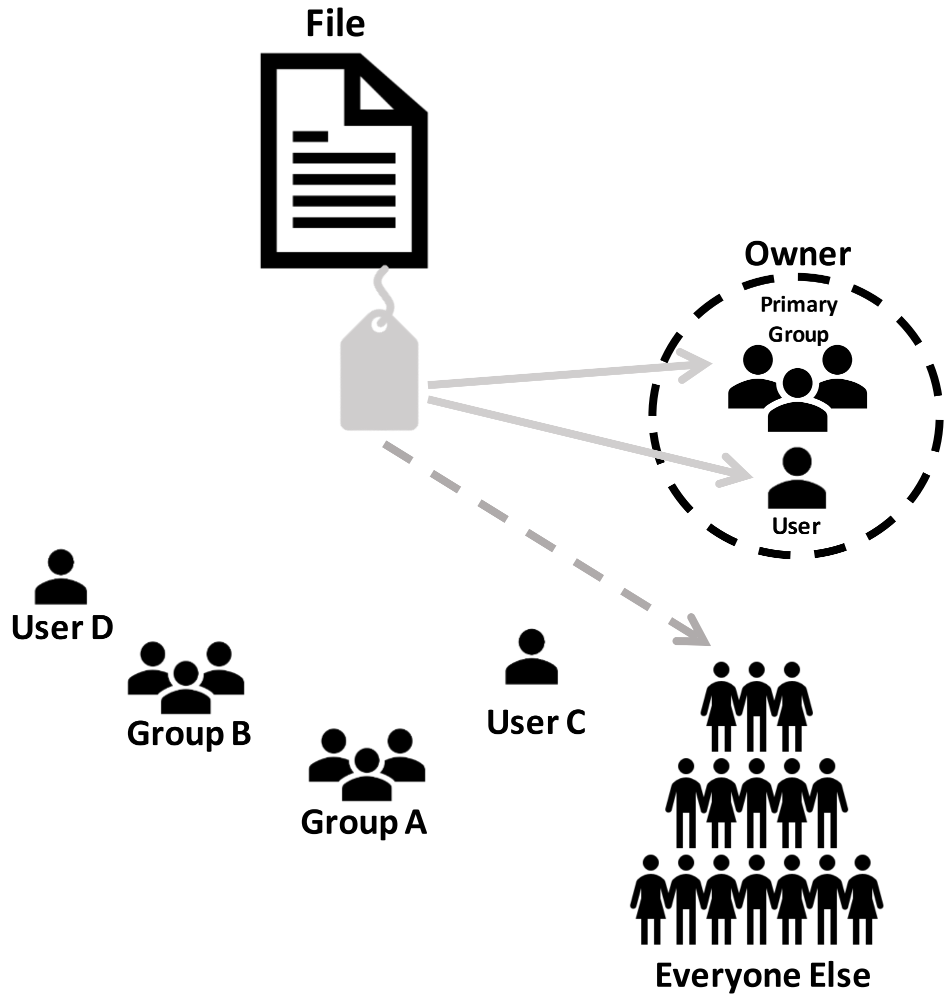
Important feature of access control based on **standard attributes** – only **one user** and only **one group** can have any rights to a file or directory.

This is a **major limitation** of the implementation.



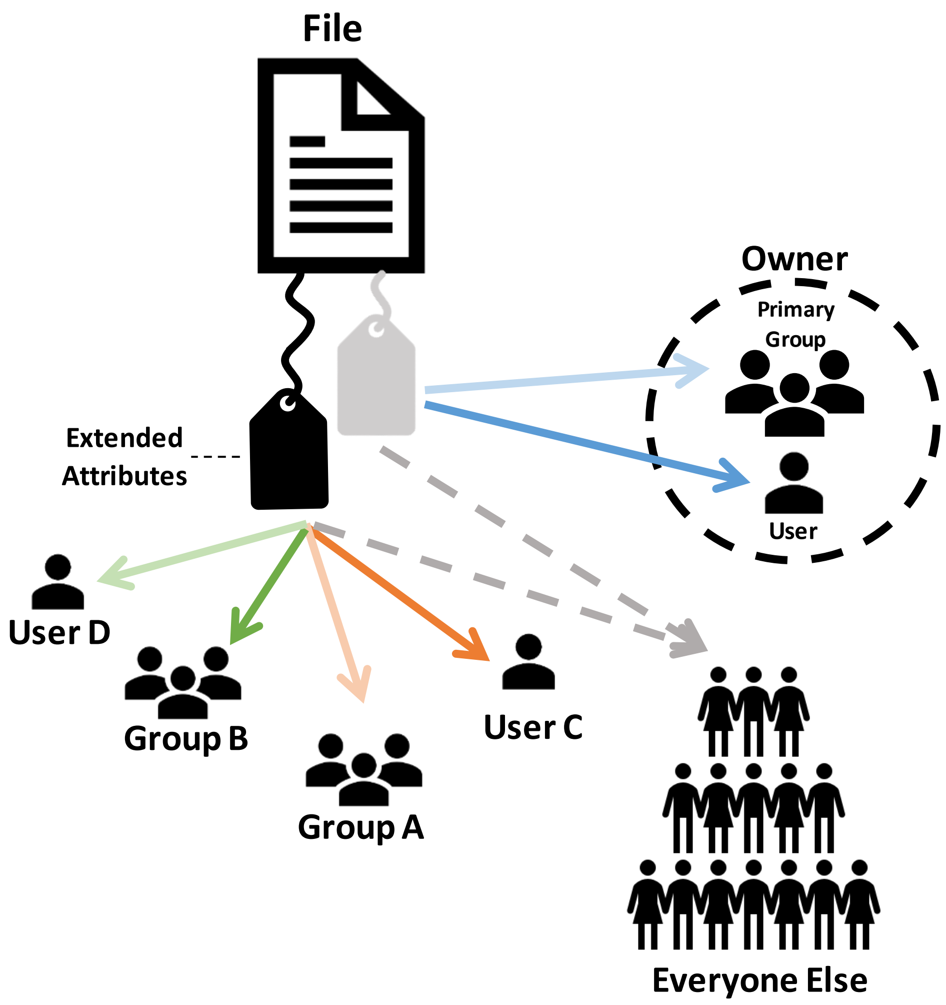
Implementation of ACL in POSIX-compliant file systems

ACL allow **enhancement** of access control by adding **expandable** list of subjects and their rights to any file or directory, creating flexible and scalable mechanism of **fine-grain control**.



Implementation of ACL in POSIX-compliant file systems

ACL allow **enhancement** of access control by adding **expandable** list of subjects and their rights to any file or directory, creating flexible and scalable mechanism of **fine-grain control**. ACL are stored in Extended Attributes of a file or a directory.

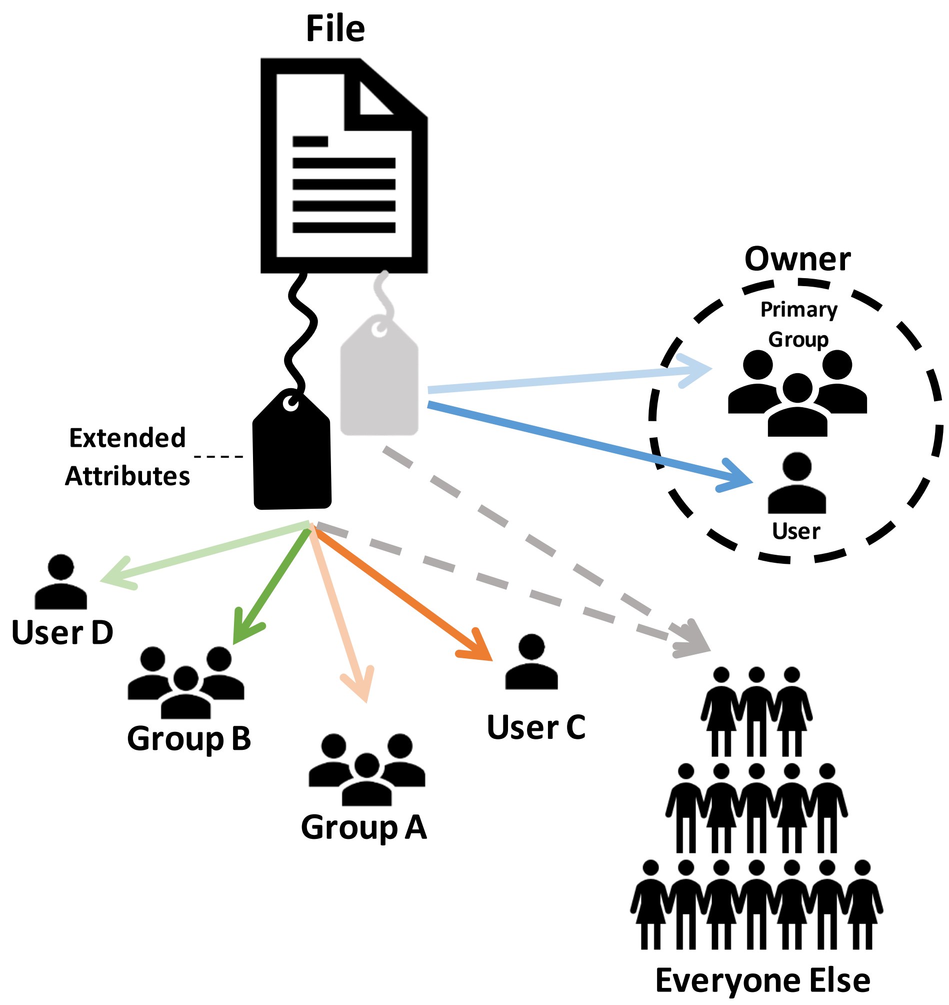


Implementation of ACL in POSIX-compliant file systems

ACL allow **enhancement** of access control by adding **expandable** list of subjects and their rights to any file or directory, creating flexible and scalable mechanism of **fine-grain control**. ACL are stored in Extended Attributes of a file or a directory.

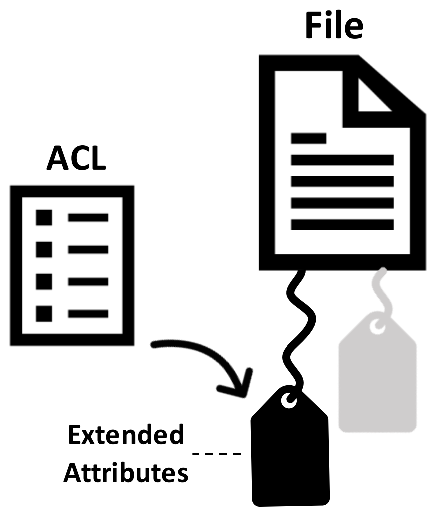
ACL are very useful in situations when:

- *number of files and directories owned by the same user require different access rights from other users*
- *number of files and directories owned by different users require shared access between them*



Implementation of ACL in POSIX-compliant file systems

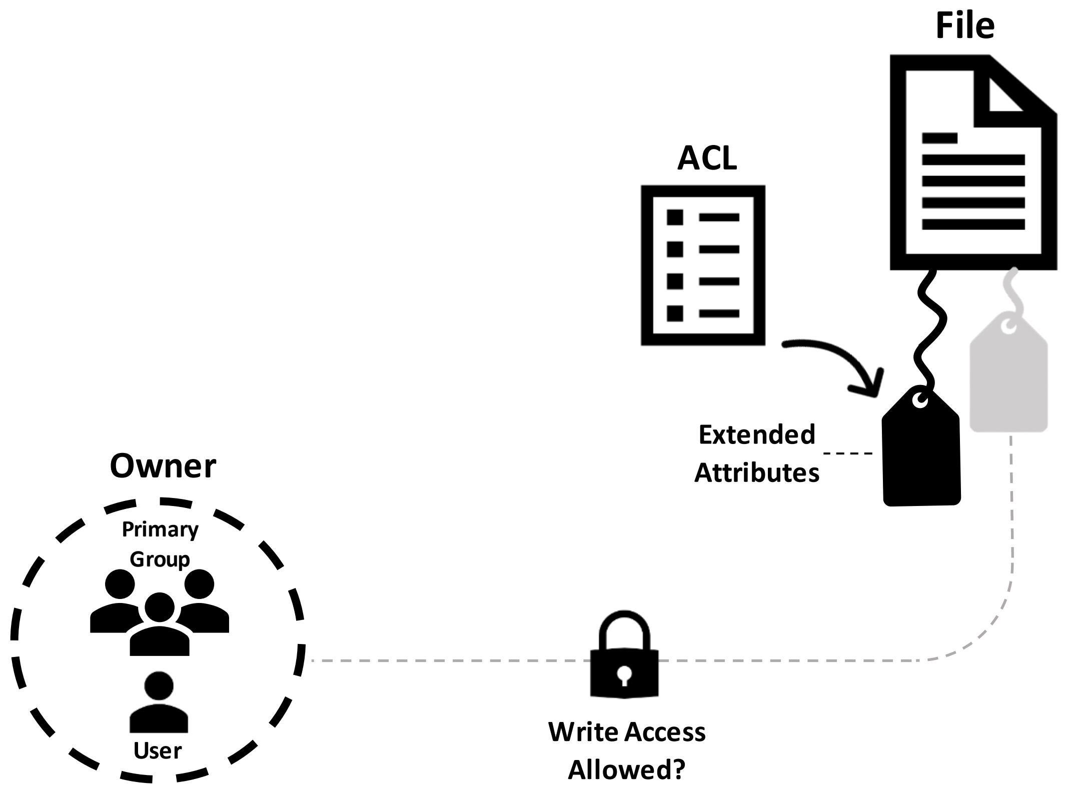
While **standard attributes** are created and assigned **automatically** at the time of creation of a file or directory, **extended attributes** must be assigned **manually**.



Implementation of ACL in POSIX-compliant file systems

While **standard attributes** are created and assigned **automatically** at the time of creation of a file or directory, **extended attributes** must be assigned **manually**.

Only **owner** with **write access** permission to a file or a directory can assign extended attributes and set up ACL. Typically, mechanism of managing ACL is implemented as a set of **commands** in operating system, specific to a file system used.



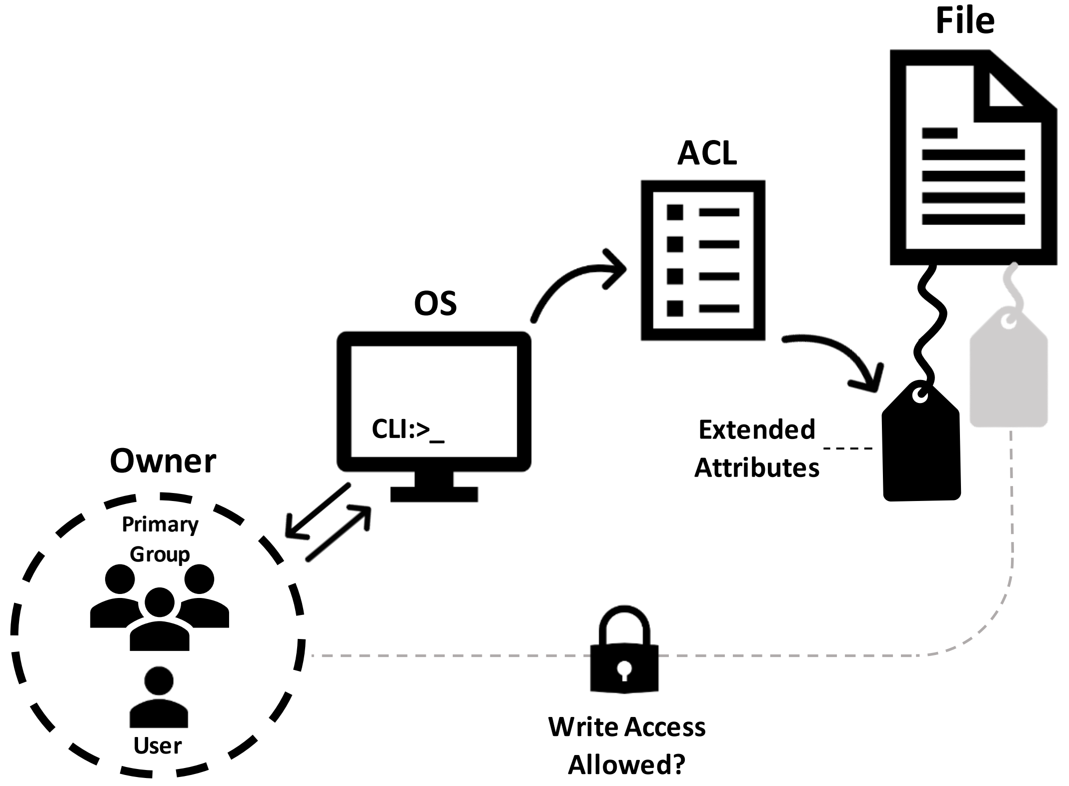
Implementation of ACL in POSIX-compliant file systems

While **standard attributes** are created and assigned **automatically** at the time of creation of a file or directory, **extended attributes** must be assigned **manually**.

Only **owner** with **write access** permission to a file or a directory can assign extended attributes and set up ACL. Typically, mechanism of managing ACL is implemented as a set of **commands** in operating system, specific to a file system used.

Subjects of ACL – user or group, **must** be **defined** on a system and **visible** to a **command** setting ACL. Also, for ACL to work, user or group listed as a subject must be defined and visible **during access** to file or directory with ACL attached.

Object of ACL – file or directory, **must** be **visible** to a **command**, setting ACL.



Setting ACL using Command Line Interface (CLI)

Typical ACL for a file or a directory can be represented as a following table:

Type	Name	Right	Flags
USER	jsmith	read, write, execute	Default
GROUP	jsmith	read, execute	Default
OTHERS		none	Default
USER	adoe	read, write	
GROUP	posixusers	read	

Setting ACL using Command Line Interface (CLI)

Typical ACL for a file or a directory can be represented as a following table:

Type	Name	Right	Flags
USER	jsmith	read, write, execute	Default
GROUP	jsmith	read, execute	Default
OTHERS		none	Default
USER	adoe	read, write	
GROUP	posixusers	read	

ACL entry with **Default** flag can be **only** created for a **directory** and, instead of setting **subject** and **access rights** for it, rather sets **default subject** and **its rights** for **any new file or directory** created **within**.

Setting ACL using Command Line Interface (CLI)

Typical ACL for a file or a directory can be represented as a following table:

Type	Name	Right	Flags
USER	jsmith	read, write, execute	Default
GROUP	jsmith	read, execute	Default
OTHERS		none	Default
USER	adoe	read, write	
GROUP	posixusers	read	

Directory

ACL entry with **Default** flag can be **only** created for a **directory** and, instead of setting **subject** and **access rights** for it, rather sets **default subject** and **its rights** for **any new file or directory** created **within**. Example:

```
[sergiy@cedar1 ~]$ cd ~/
[sergiy@cedar1 ~]$ mkdir webinar
```

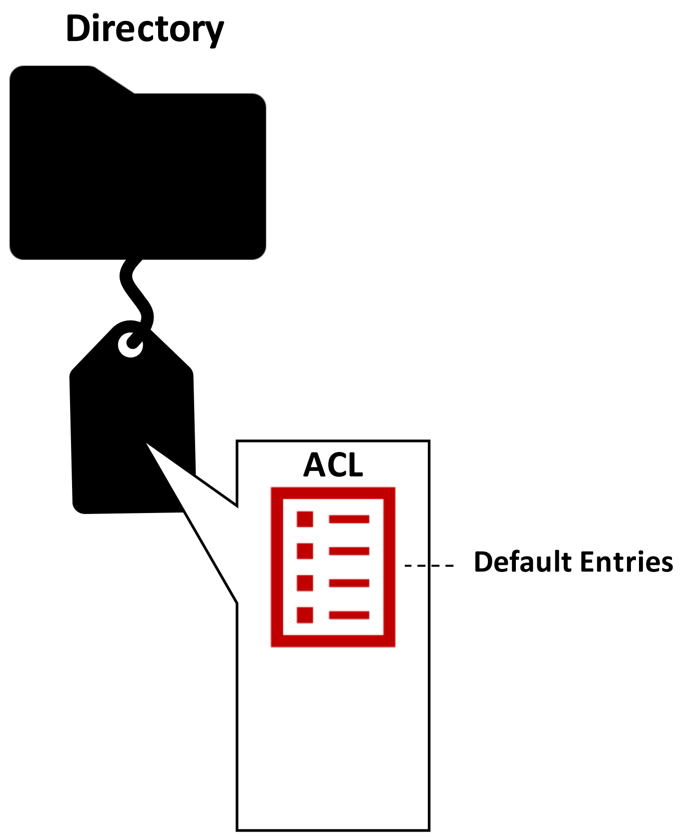
Setting ACL using Command Line Interface (CLI)

Typical ACL for a file or a directory can be represented as a following table:

Type	Name	Right	Flags
USER	jsmith	read, write, execute	Default
GROUP	jsmith	read, execute	Default
OTHERS		none	Default
USER	adoe	read, write	
GROUP	posixusers	read	

ACL entry with **Default** flag can be **only** created for a **directory** and, instead of setting **subject** and **access rights** for it, rather sets **default subject** and **its rights** for **any new file or directory** created **within**. Example:

```
[sergiy@cedar1 ~]$ cd ~/
[sergiy@cedar1 ~]$ mkdir webinar
[sergiy@cedar1 ~]$ setfacl -d -m g:wg_staff:rx webinar
```



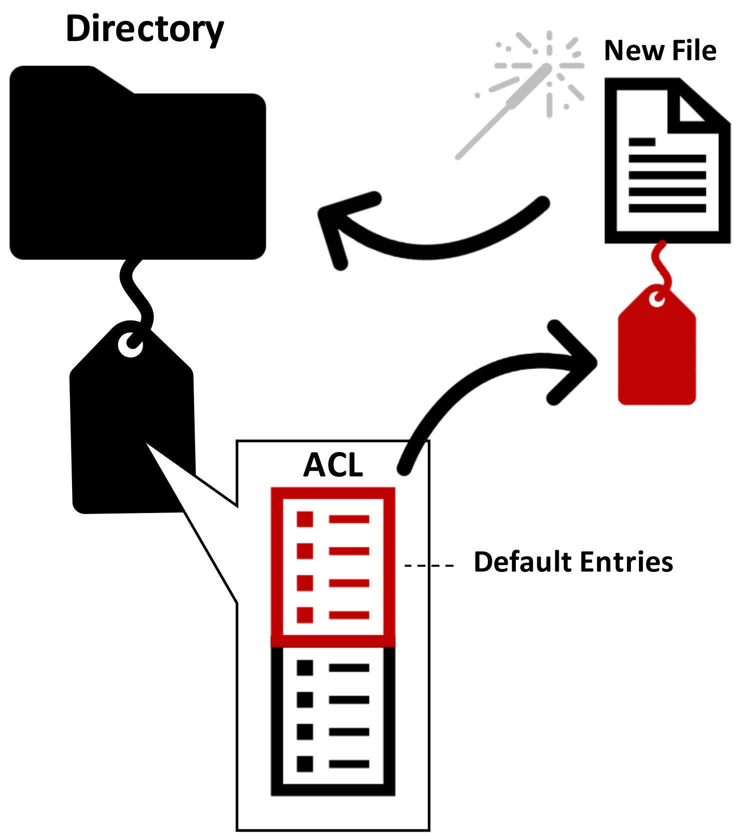
Setting ACL using Command Line Interface (CLI)

Typical ACL for a file or a directory can be represented as a following table:

Type	Name	Right	Flags
USER	jsmith	read, write, execute	Default
GROUP	jsmith	read, execute	Default
OTHERS		none	Default
USER	adoe	read, write	
GROUP	posixusers	read	

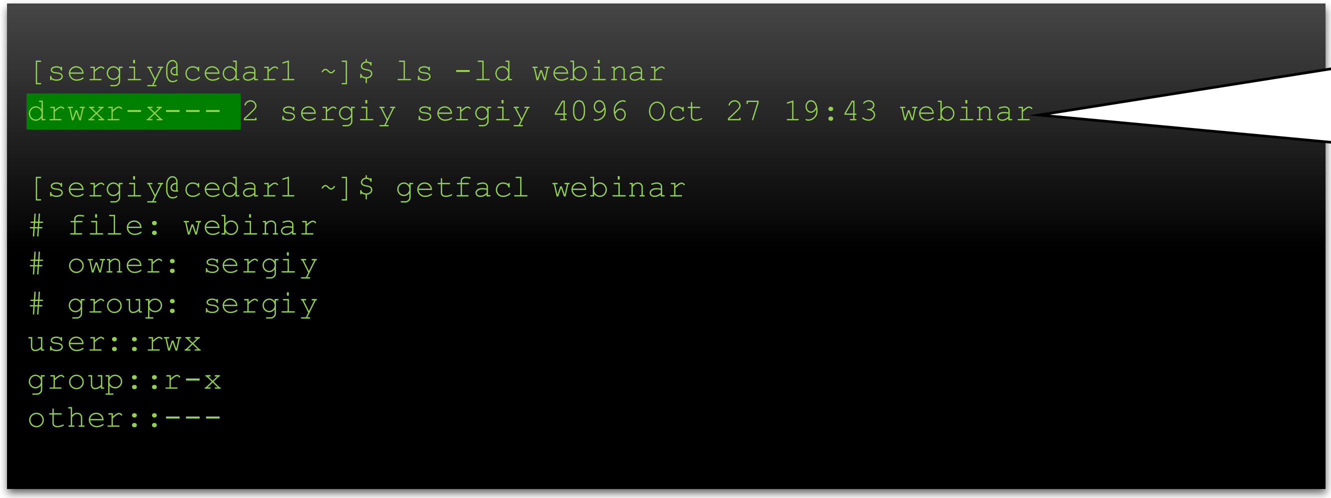
ACL entry with **Default** flag can be **only** created for a **directory** and, instead of setting **subject** and **access rights** for it, rather sets **default subject** and **its rights** for **any new file or directory** created **within**. Example:

```
[sergiy@cedar1 ~]$ cd ~/
[sergiy@cedar1 ~]$ mkdir webinar
[sergiy@cedar1 ~]$ setfacl -d -m g:wg_staff:rx webinar
[sergiy@cedar1 ~]$ setfacl -m g:wg_staff:rx webinar
[sergiy@cedar1 ~]$ touch webinar/sample-0027.file
```



Setting ACL using Command Line Interface (CLI)

ACL entries obey restrictions set by permissions within standard attributes of a file or a directory. For example: we have just created directory called webinar. If we would check its standard permissions before applying ACL we would find following:



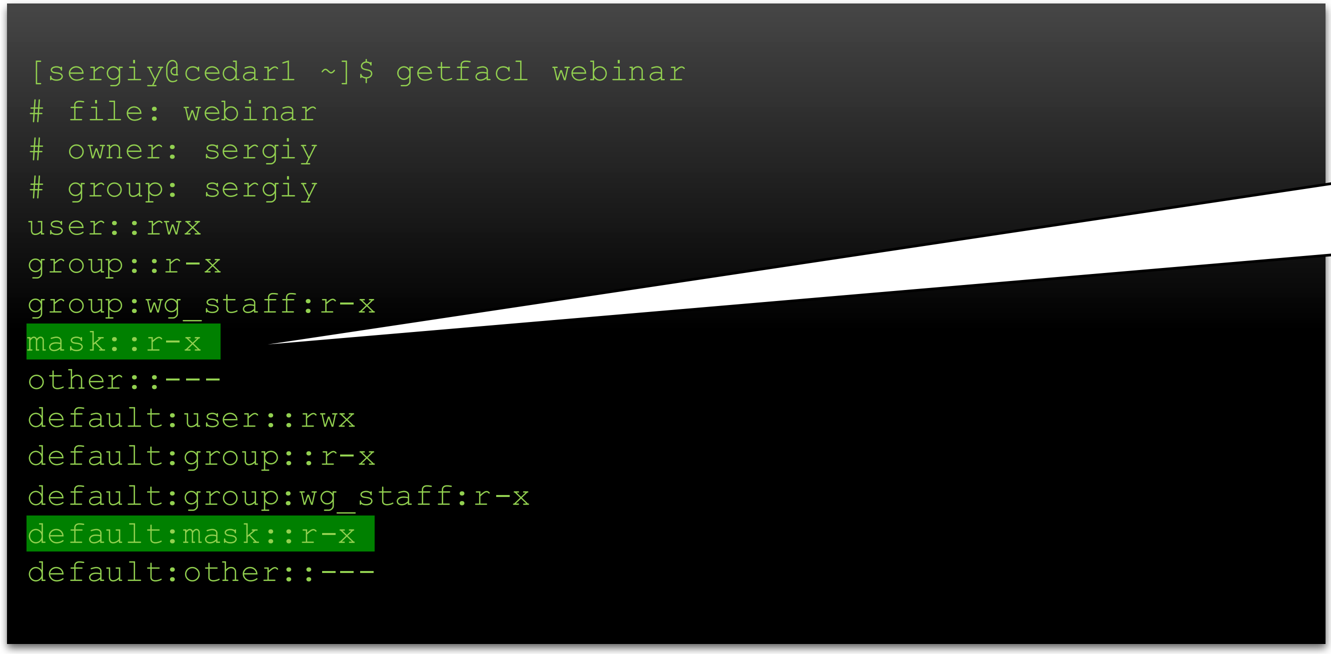
```
[sergiy@cedar1 ~]$ ls -ld webinar
drwxr-x--- 2 sergiy sergiy 4096 Oct 27 19:43 webinar

[sergiy@cedar1 ~]$ getfacl webinar
# file: webinar
# owner: sergiy
# group: sergiy
user::rwx
group::r-x
other::---
```

$$\mathtt{d}\ \underbrace{\mathtt{rwx}}_{1}\ \underbrace{\mathtt{r-x}}_{2}\ \underbrace{\mathtt{---}}_{3}$$

1. Primary user can read, write and enter
2. Primary group can read and enter
3. Others can not have any access

Setting ACL using Command Line Interface (CLI)

After applying ACL in earlier slides we allowed additional group **wg_staff** to **read** and **enter** the directory and set up this rule as a **default** for **webinar** directory, so all new files and directories in it will inherit the same access: **wg_staff** will be able to **read files** and **read/enter directories**:



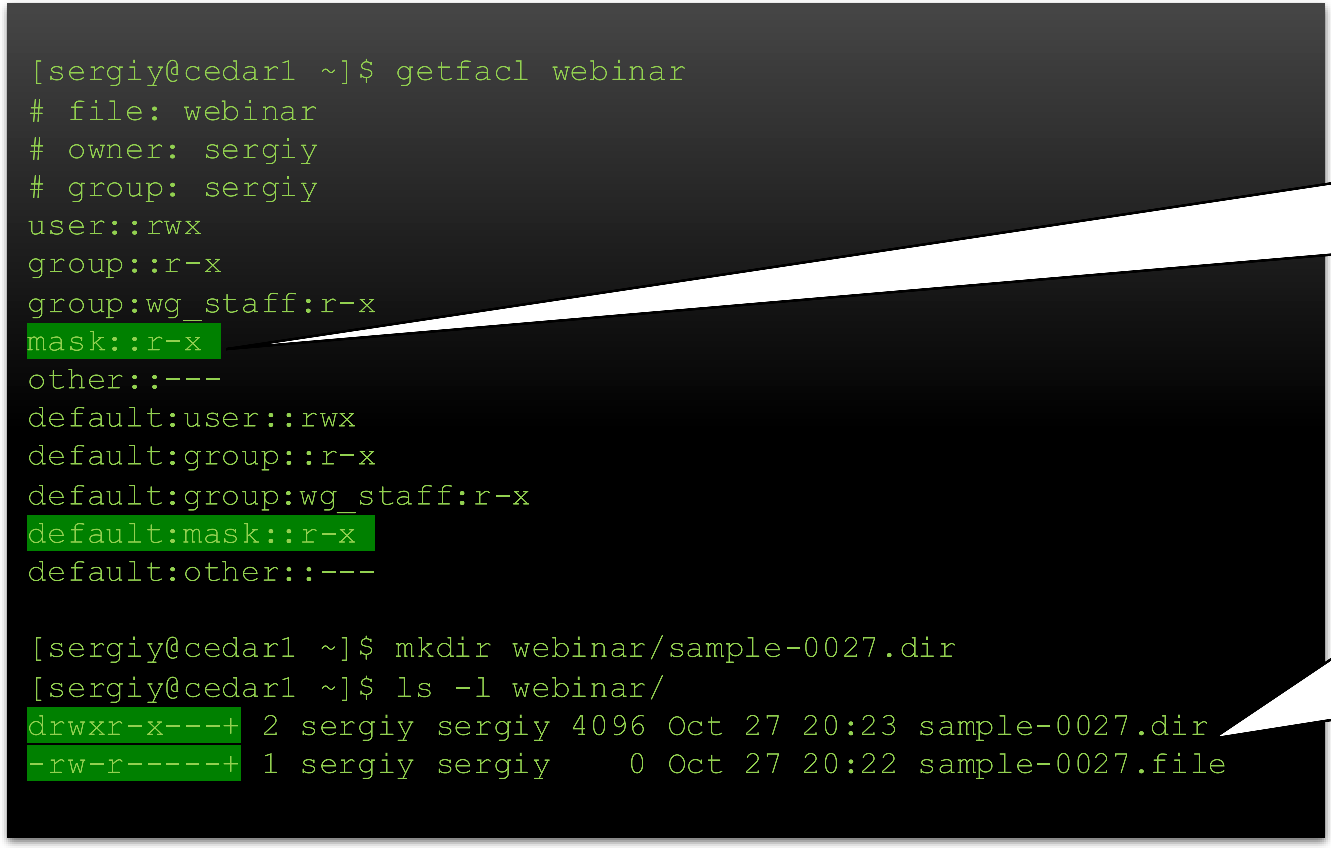
```
[sergiy@cedar1 ~]$ getfacl webinar
# file: webinar
# owner: sergiy
# group: sergiy
user::rwx
group::r-x
group:wg_staff:r-x
mask::r-x
other::---
default:user::rwx
default:group::r-x
default:group:wg_staff:r-x
default:mask::r-x
default:other::---
```

`mask::r-x`

Combination of standard attribute permissions and newly set up permissions by ACL.

Setting ACL using Command Line Interface (CLI)

After applying ACL in earlier slides we allowed additional group **wg_staff** to **read** and **enter** the directory and set up this rule as a **default** for **webinar** directory, so all new files and directories in it will inherit the same access: **wg_staff** will be able to **read files** and **read/enter directories**:



```
[sergiy@cedar1 ~]$ getfacl webinar
# file: webinar
# owner: sergiy
# group: sergiy
user::rwx
group::r-x
group:wg_staff:r-x
mask::r-x
other::---
default:user::rwx
default:group::r-x
default:group:wg_staff:r-x
default:mask::r-x
default:other::---

[sergiy@cedar1 ~]$ mkdir webinar/sample-0027.dir
[sergiy@cedar1 ~]$ ls -l webinar/
drwxr-x---+ 2 sergiy sergiy 4096 Oct 27 20:23 sample-0027.dir
-rw-r-----+ 1 sergiy sergiy    0 Oct 27 20:22 sample-0027.file
```

`mask::r-x`

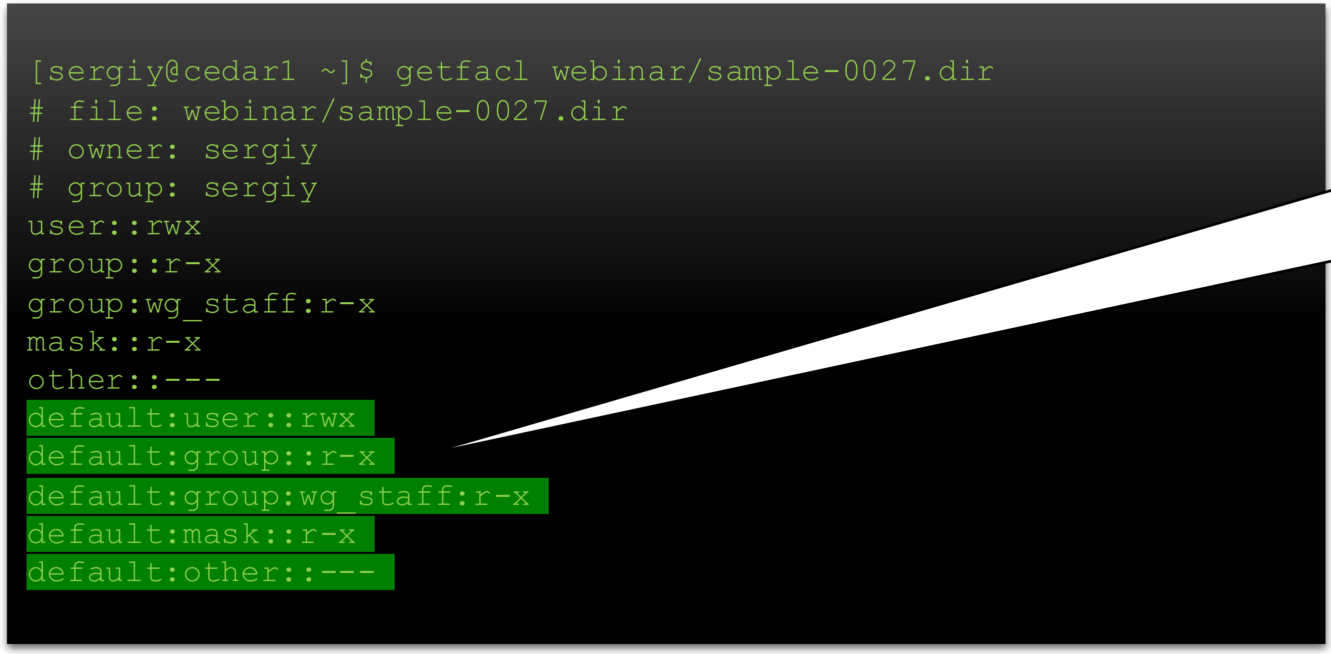
Combination of standard attribute permissions and newly set up permissions by ACL.

+ sign at the end of permission attributes indicates presence of ACL.

`sample-0027.dir` and **`sample-0027.file`** have ACL automatically set up from default entries of **`webinar`**.

Setting ACL using Command Line Interface (CLI)

As long as **default** ACL entries set up – all **new data** will inherit it automatically and pass it along to **new data nested** within:



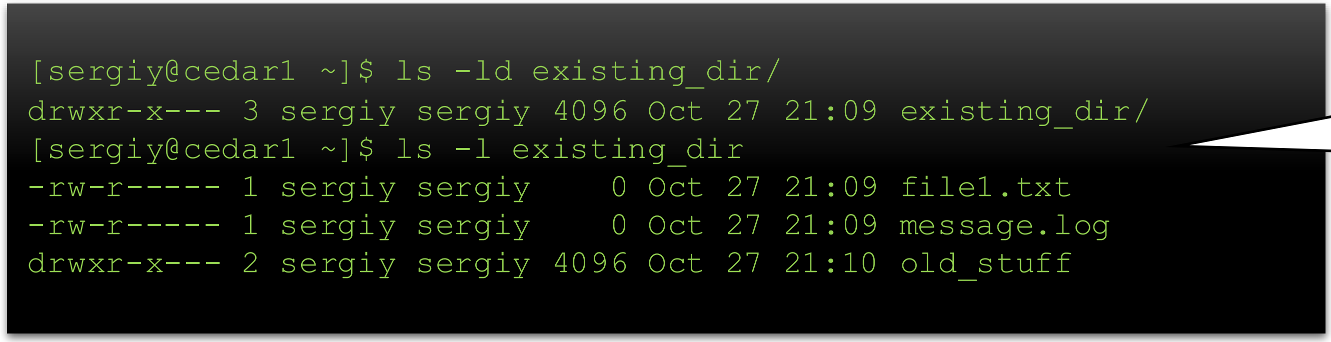
```
[sergiy@cedar1 ~]$ getfacl webinar/sample-0027.dir
# file: webinar/sample-0027.dir
# owner: sergiy
# group: sergiy
user::rwx
group::r-x
group:wg_staff:r-x
mask::r-x
other::---
default:user::rwx
default:group::r-x
default:group:wg_staff:r-x
default:mask::r-x
default:other::---
```

All default ACL entries from parent directory webinar are inherited by this directory and will be passed along to any new file or directory within itself. This process will continue in perpetuity.

It will not, however, affect **existing** files or directories. It has to be done directly with **regular** ACL entries. I use commands **setfacl** and **getfacl** as standard tools available in all major Linux distributions. Let's examine how direct setting of ACL is done.

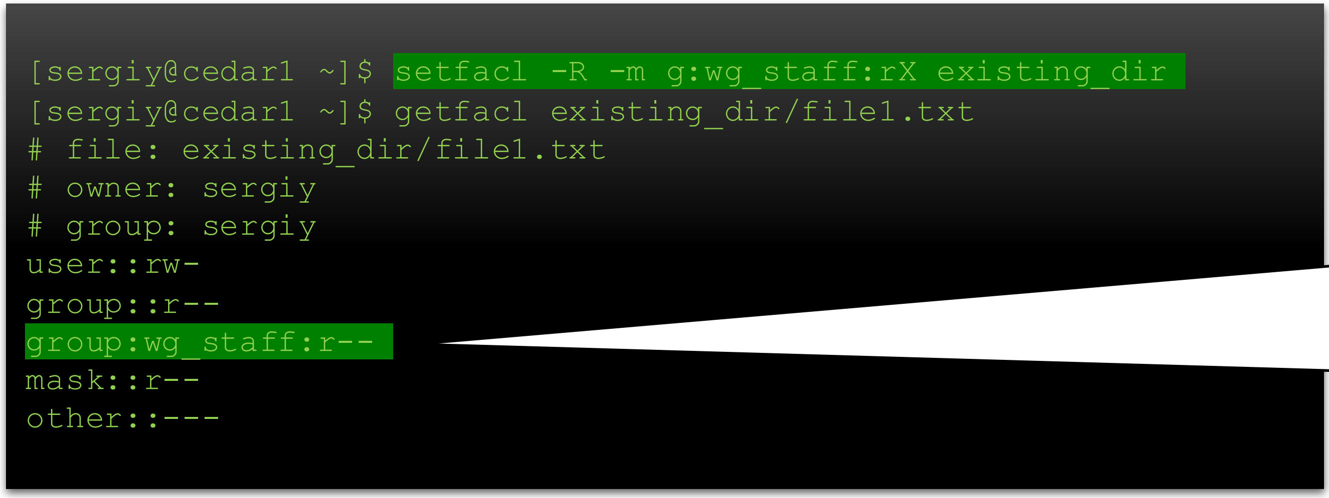
Setting ACL using Command Line Interface (CLI)

I have directory called **existing_dir** without any ACL attached to it and with some files and directories inside:



```
[sergiy@cedar1 ~]$ ls -ld existing_dir/
drwxr-x--- 3 sergiy sergiy 4096 Oct 27 21:09 existing_dir/
[sergiy@cedar1 ~]$ ls -l existing_dir
-rw-r----- 1 sergiy sergiy    0 Oct 27 21:09 file1.txt
-rw-r----- 1 sergiy sergiy    0 Oct 27 21:09 message.log
drwxr-x--- 2 sergiy sergiy 4096 Oct 27 21:10 old_stuff
```

Parent directory and everything within it can be only accessed by me and my primary group



```
[sergiy@cedar1 ~]$ setfacl -R -m g:wg_staff:rX existing_dir
[sergiy@cedar1 ~]$ getfacl existing_dir/file1.txt
# file: existing_dir/file1.txt
# owner: sergiy
# group: sergiy
user::rw-
group::r--
group:wg_staff:r--
mask::r--
other::---
```

After running **setfacl** recursively on the entire **`existing_dir`**, all files and directories now have ACL entries allowing **wg_staff** to read files and read/enter directories. It is achieved by using **`X`** in access rights and *-R* parameter in the command.

`g:wg_staff:rX`

ACL Best Practices and Use Cases

Keeping access control to data well-organized and maintained is a foundation of information security and an indicator of mature data management. However, data sets increase in size, number of files and directories we deal regularly exceeds millions on daily basis and number of people required special access to portion of our data grows constantly. How to scale-out management overhead for access control?

- **Use default ACL entries as much as possible – let computer do all hard work of propagating ACL to new data**
- **Use group-based ACL entries as much as possible – manage group members instead of changing ACL again and again**
- **Keep record of all implemented ACL – habit, that will safe you a lot of time if and when you need to recreate ACL entries from scratch**
- **Use access control management automation – write your own or get ready-to use tools**

ACL Best Practices and Use Cases

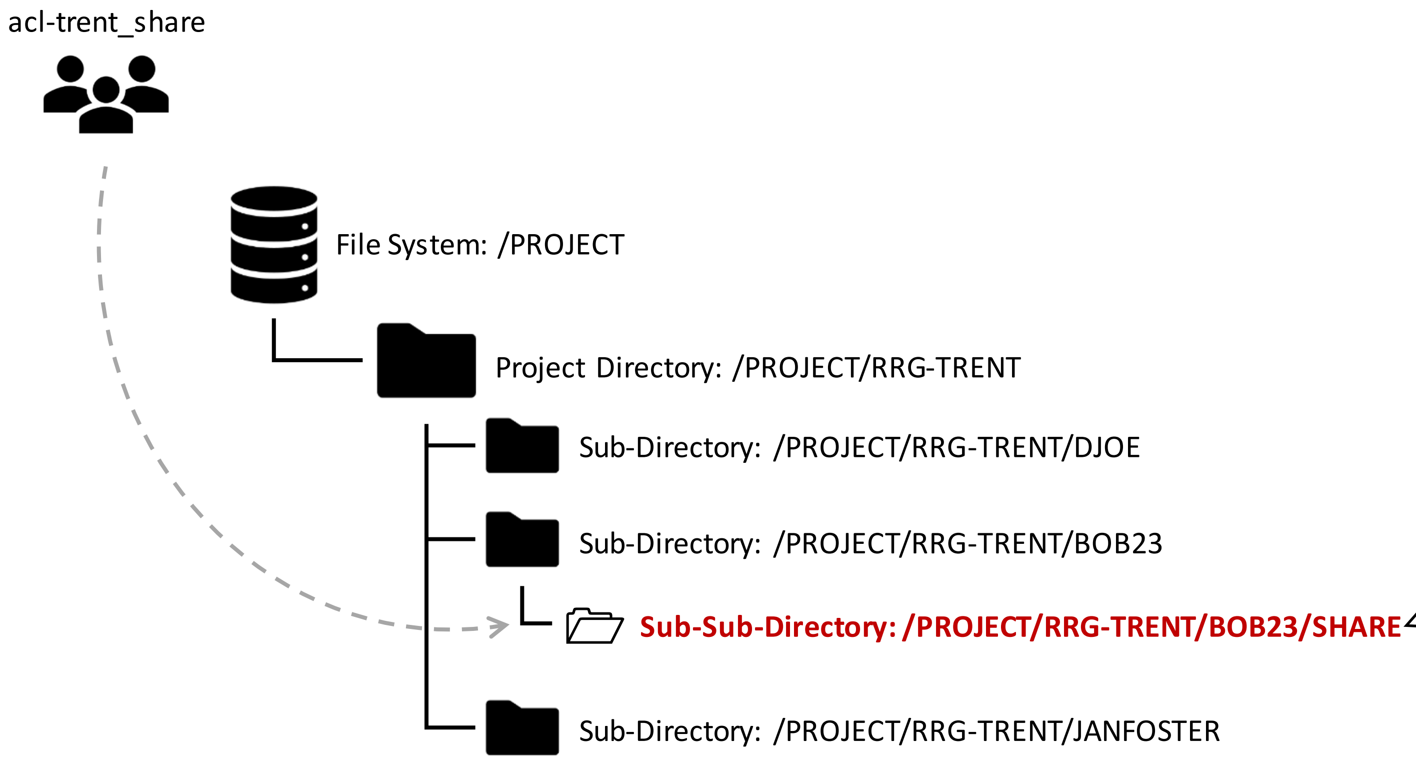
Group-based ACL entries are one of the most **powerful** tools for access control you can have, On your own system – you can manage groups for that. On **shared** systems like Compute Canada's **CEDAR**, **GRAHAM** and others – you need help of **support personnel** to create such groups. Follow three simple steps to setup group-based ACL:

1. Send email to support@computecanada.ca with a request to create CCDB group for data sharing. Provide desired name of the group, name of a person who will manage the group (can be yourself) and names of group members (optional). Note, that request must be **approved by PI** if you are a sponsored user.
2. Upon receiving response from Compute Canada's support about creation of the group, either yourself or a designated manager should login to group management interface of CCDB at https://ccdb.computecanada.ca/services. All groups available for management will be displayed. Add or remove users to the group you need.
3. Via SSH-client login to Compute Canada's cluster where you would like to set up ACL. Use command line to find directory or file (less effective, but acceptable) you wish to set up ACL and run command ***setfacl***, using group's name in ACL parameters

⚠ Please remember, that you can only set up ACL on files and directories you have write permissions to.

ACL Best Practices and Use Cases

Use case: Access to Directory inside Directory Tree


acl-trent_share
File System: /PROJECT
Project Directory: /PROJECT/RRG-TRENT
Sub-Directory: /PROJECT/RRG-TRENT/DJOE
Sub-Directory: /PROJECT/RRG-TRENT/BOB23
Sub-Sub-Directory: /PROJECT/RRG-TRENT/BOB23/SHARE
Sub-Directory: /PROJECT/RRG-TRENT/JANFOSTER

We would like to allow **read-only** access to directory **SHARE** inside of **/PROJECT/RRG-TRENT** directory tree for members of the group called **acl-trent_share**.

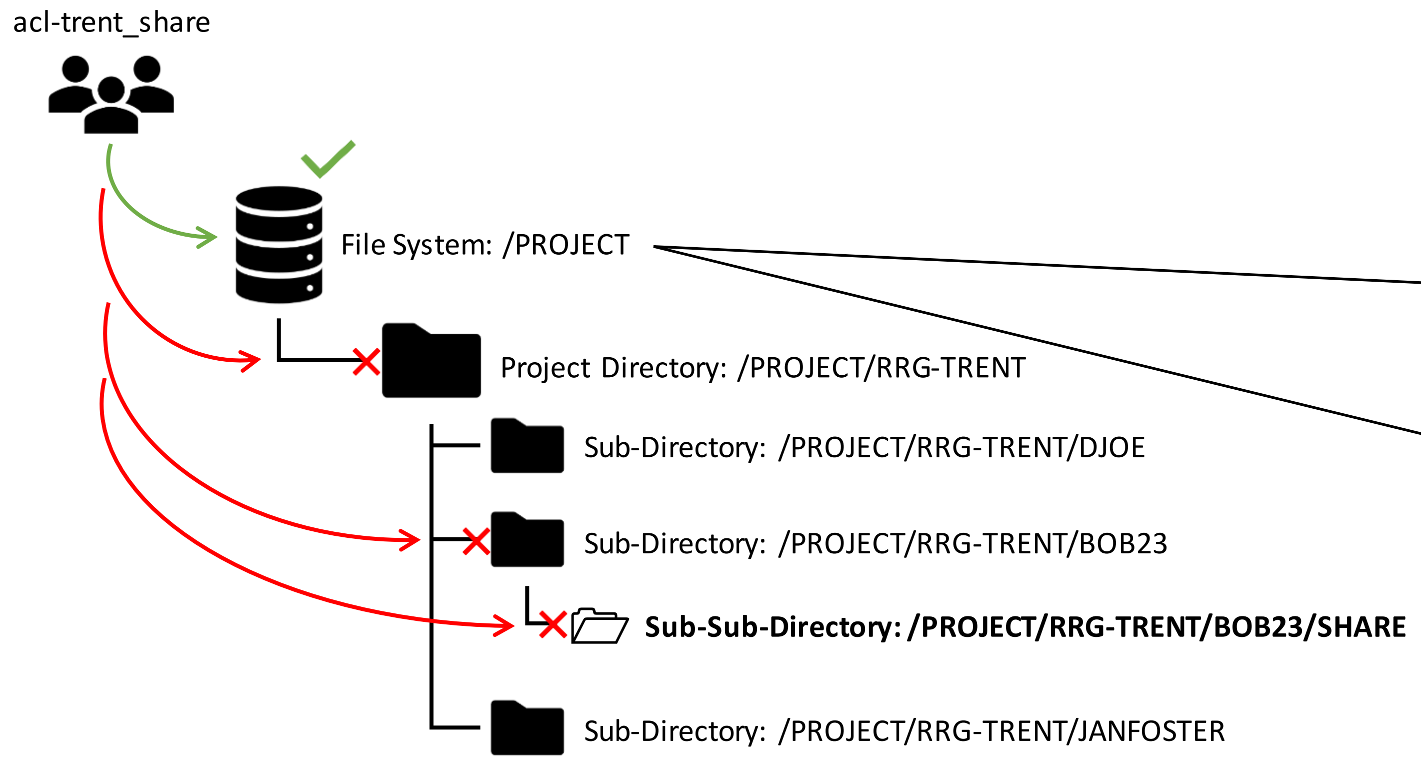
User BOB23 owns directories SHARE and BOB23, but /PROJECT/RRG-TRENT is own by user TRENT. Both users TRENT and BOB23 are members of group RRG-TRENT.

Group RRG-TRENT has no access to any subdirectories within /PROJECT/RRG-TRENT, but can access /PROJECT/RRG-TRENT itself.

Nobody else can access /PROJECT/RRG-TRENT

ACL Best Practices and Use Cases

Use case: Access to Directory inside Directory Tree

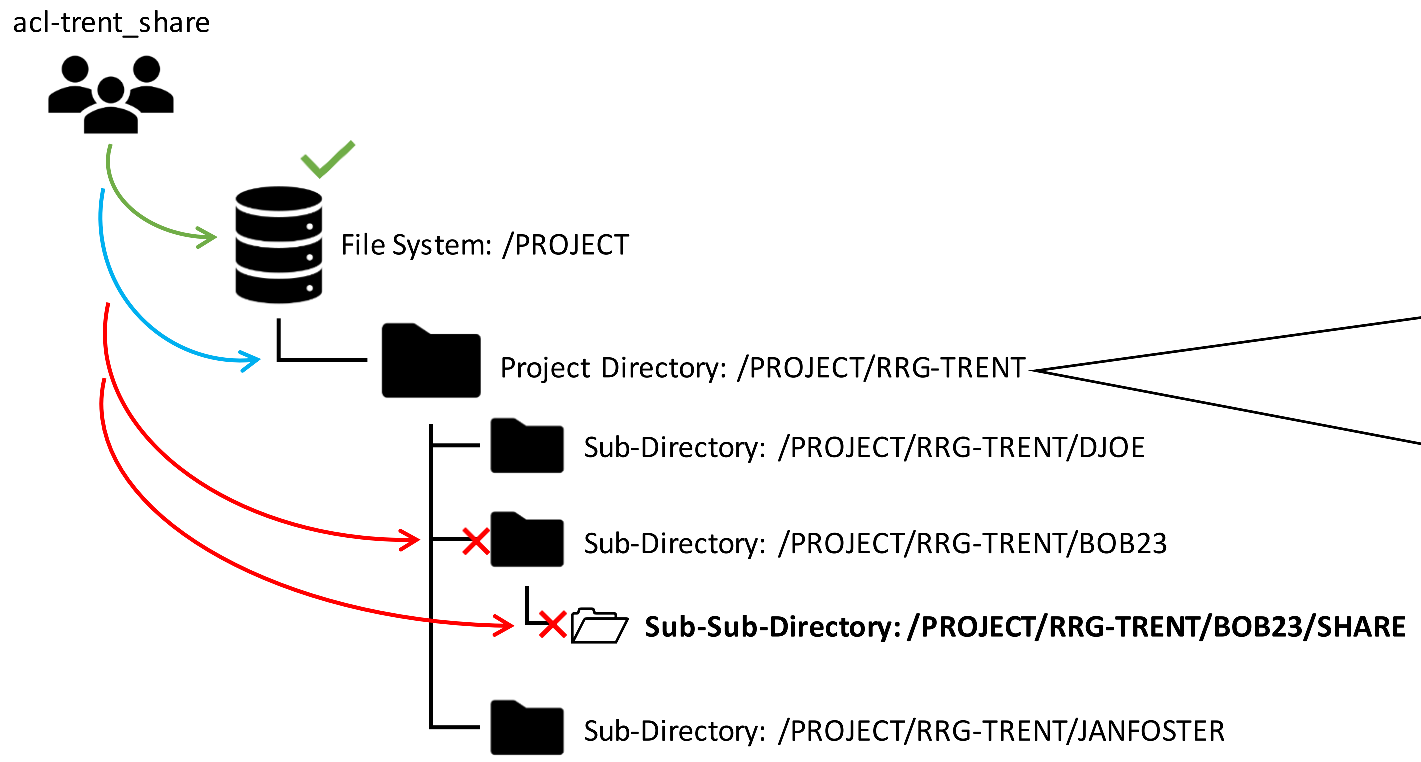

acl-trent_share
File System: /PROJECT
Project Directory: /PROJECT/RRG-TRENT
Sub-Directory: /PROJECT/RRG-TRENT/DJOE
Sub-Directory: /PROJECT/RRG-TRENT/BOB23
Sub-Sub-Directory: /PROJECT/RRG-TRENT/BOB23/SHARE
Sub-Directory: /PROJECT/RRG-TRENT/JANFOSTER

Group **acl-trent_share** can access file system /PROJECT, as it is accessible to all known users on the system.

Directory /PROJECT/RRG-TRENT and its entire content are not accessible by the group.

ACL Best Practices and Use Cases

Use case: Access to Directory inside Directory Tree


acl-trent_share
File System: /PROJECT
Project Directory: /PROJECT/RRG-TRENT
Sub-Directory: /PROJECT/RRG-TRENT/DJOE
Sub-Directory: /PROJECT/RRG-TRENT/BOB23
Sub-Sub-Directory: /PROJECT/RRG-TRENT/BOB23/SHARE
Sub-Directory: /PROJECT/RRG-TRENT/JANFOSTER

Group **acl-trent_share** can access file system /PROJECT, as it is accessible to all known users on the system.

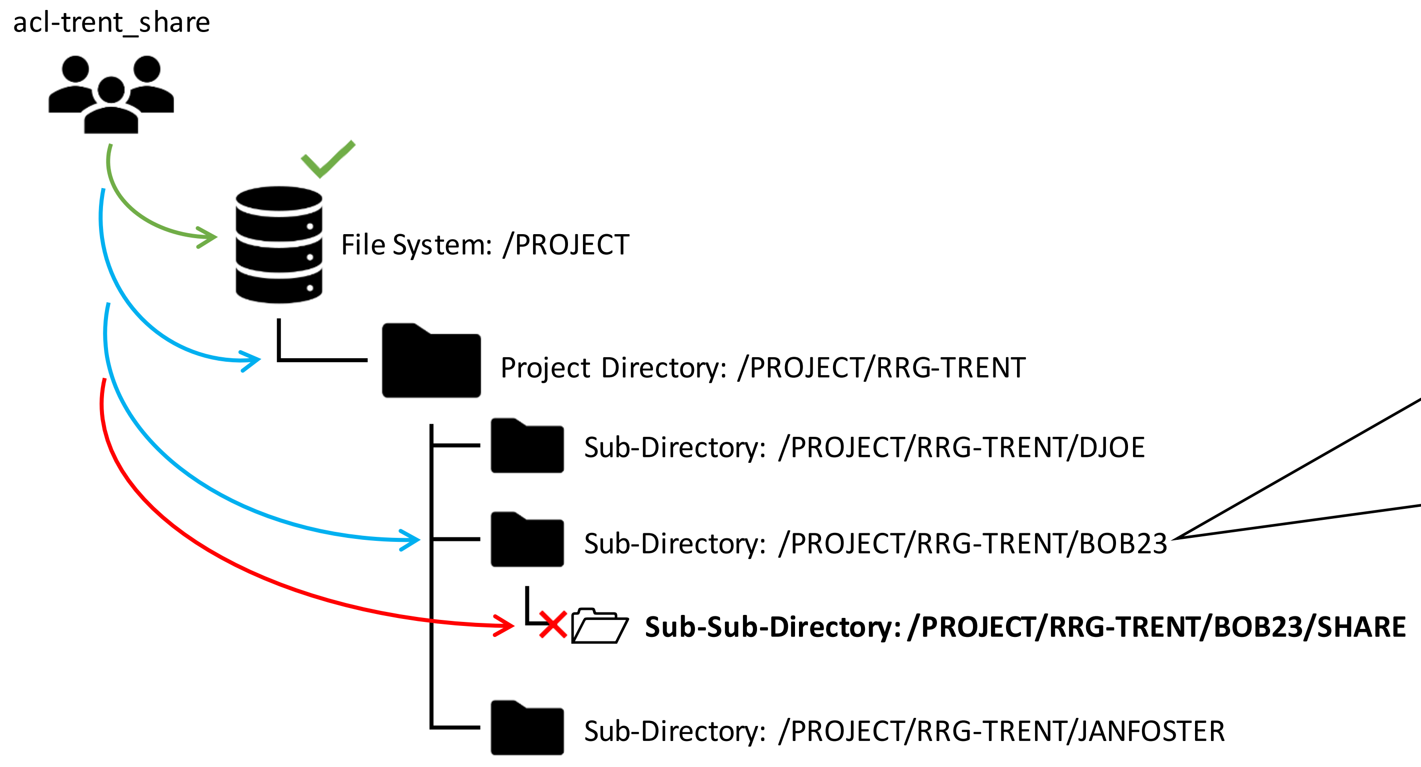
Directory /PROJECT/RRG-TRENT and its entire content are not accessible by the group.

Either user TRENT or system administrator, upon request from user TRENT can allow **enter** type of access to directory /PROJECT/RRG-TRENT for **everybody**. It will NOT allow everybody to **list** content of the directory or **read** its content.

```
[sergiy@cedar1 ~]$ chmod o+X /project/rrg-trent/
```

ACL Best Practices and Use Cases

Use case: Access to Directory inside Directory Tree

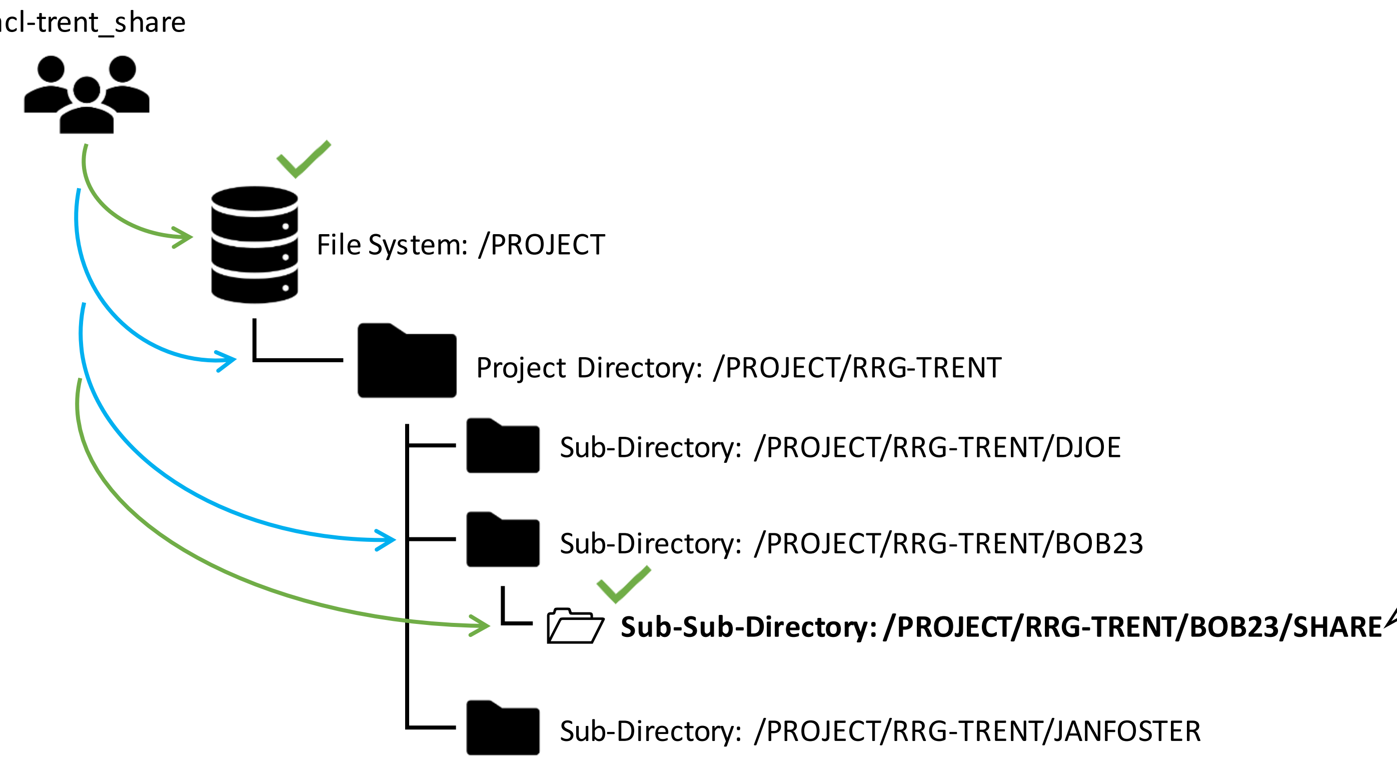

acl-trent_share
File System: /PROJECT
Project Directory: /PROJECT/RRG-TRENT
Sub-Directory: /PROJECT/RRG-TRENT/DJOE
Sub-Directory: /PROJECT/RRG-TRENT/BOB23
Sub-Sub-Directory: /PROJECT/RRG-TRENT/BOB23/SHARE
Sub-Directory: /PROJECT/RRG-TRENT/JANFOSTER

Either user BOB23 or system administrator, upon request from user BOB23 can allow **enter** type of access to directory /PROJECT/RRG-TRENT/BOB23 for **everybody**. It will NOT allow everybody to **list** content of the directory or **read** its content.

```
[sergiy@cedar1 ~]$ chmod o+X /project/rrg-trenty/bob23
```

ACL Best Practices and Use Cases

Use case: Access to Directory inside Directory Tree


acl-trent_share
File System: /PROJECT
Project Directory: /PROJECT/RRG-TRENT
Sub-Directory: /PROJECT/RRG-TRENT/DJOE
Sub-Directory: /PROJECT/RRG-TRENT/BOB23
Sub-Sub-Directory: /PROJECT/RRG-TRENT/BOB23/SHARE
Sub-Directory: /PROJECT/RRG-TRENT/JANFOSTER

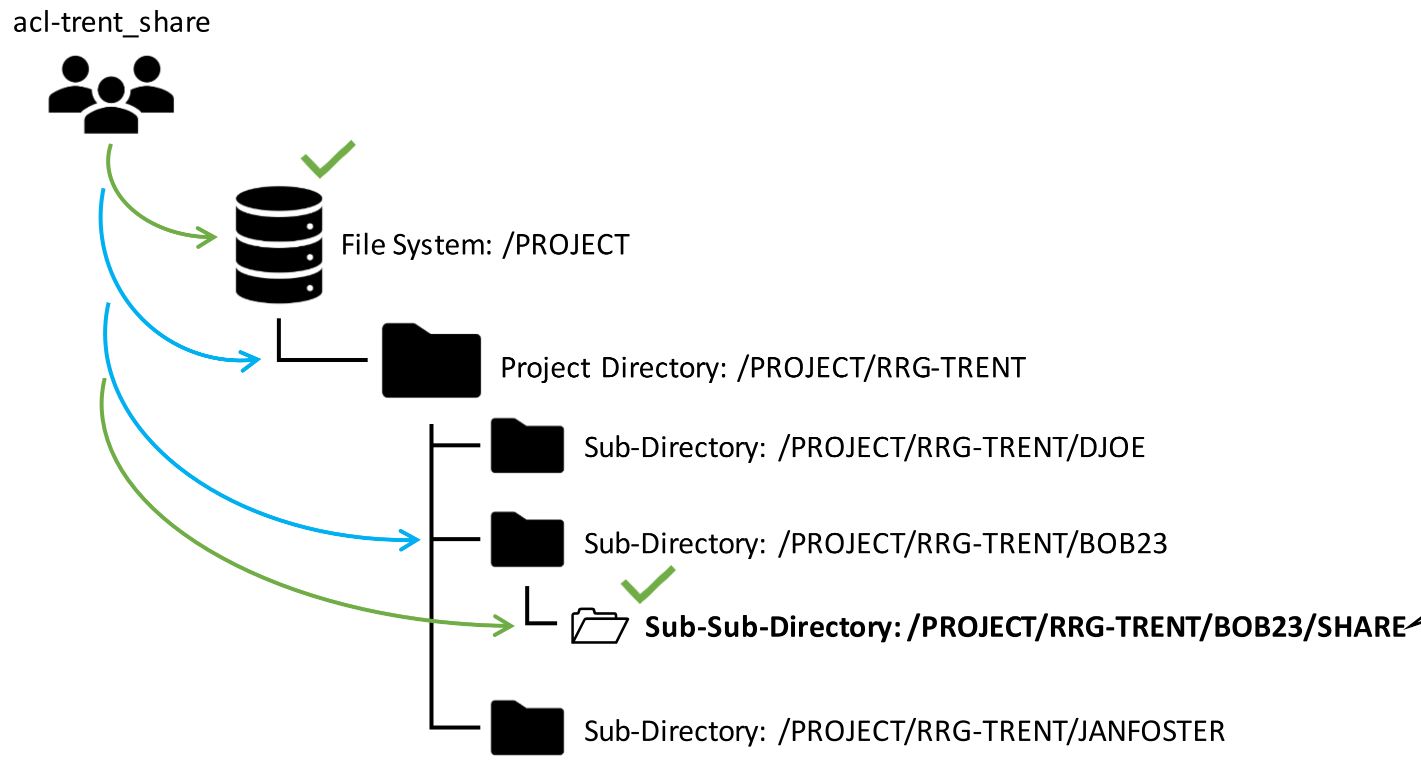
Either user BOB23 or system administrator, upon request from user BOB23 can allow **enter** type of access to directory /PROJECT/RRG-TRENT/BOB23 for **everybody**. It will NOT allow everybody to **list** content of the directory or **read** its content.

User BOB23 can can set up ACL with **setfacl** command to allow read-only access to /PROJECT/RRG-TRENT/BOB23/SHARE to specific group **acl-trent_share**. Read-only access implies enter type of access as well. –R parameter in setfacl allows to apply the ACL to entire content of the directory

```
[sergiy@cedar1 ~]$ setfacl -R -m g:acl-trent_share:rx /project/rrg-trent/bob23/share
```

ACL Best Practices and Use Cases

Use case: Access to Directory inside Directory Tree


acl-trent_share
File System: /PROJECT
Project Directory: /PROJECT/RRG-TRENT
Sub-Directory: /PROJECT/RRG-TRENT/DJOE
Sub-Directory: /PROJECT/RRG-TRENT/BOB23
Sub-Sub-Directory: /PROJECT/RRG-TRENT/BOB23/SHARE
Sub-Directory: /PROJECT/RRG-TRENT/JANFOSTER

To make sure that new files and directories in /PROJECT/RRG-TRENT/BOB23/SHARE are also readable by the acl-trent_share group – default ACL entry needs to be set with **–d** parameter of setfacl command

```
[sergiy@cedar1 ~]$ setfacl -d -m g:acl-trent_share:rx /project/rrg-trent/bob23/share
```