Programming in shell 1

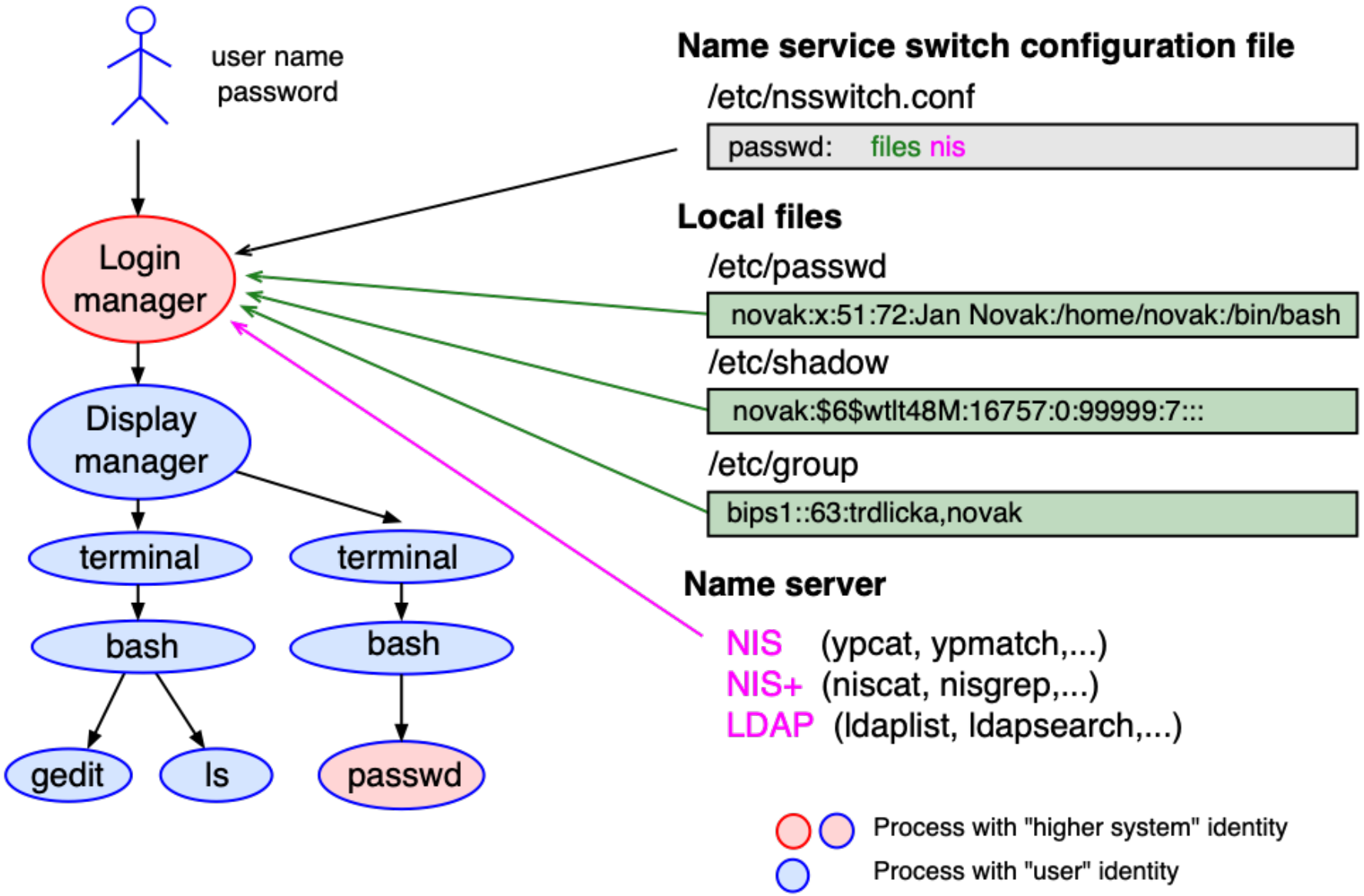
Process identity. Access permissions.

Jan Trdlička

Czech Technical University in Prague, Faculty of Information Technology
Department of Computer Systems

Content

System login



User account

- A user account must be created before the user can log on.
- The following informations represent the user account.
  - User name
  - User ID: *UID*
    - $UID = 0$ represents admin account (root account).
  - Primary group ID: GID_1
    - User belongs to at least one primary group.
  - List of secondary group ID's: GID_2, GID_3, ... , GID_n (optional)
    - User can set his secondary to primary group by command `newgrp`.
  - Comment
    - Description of the user account.
  - Home directory
    - Working directory is set to home directory after login.
  - Login shell
  - User password

User account database

- The information defined within the user account can be stored in different locations.
- Name service switch configuration file
  - `/etc/nsswitch.conf`
  - Defines where the information about user accounts are saved.
  - This information can be get by command `getent`.
- Local Files
  - `/etc/passwd`
  - `/etc/shadow`
  - `/etc/group`
- Central name service
  - Keeps information about user accounts at one place (not only about user accounts ...).
  - NIS (commands `ypcat`, `ypmatch`, ...)
  - NIS+ (commands `niscat`, `nisgrep`, ...)
  - LDAP (commands `ldaplist`, `ldapsearch`, ...)

User account database – local file structure

- File /etc/passwd

  name:x:UID:GID_1:comment:home_directory:login_shell

- File /etc/shadow

  name:password:lastchg:min:max:warn:inactive:expire:flag

- File /etc/group

  group::GID_i:list_of_users

Login process

1. System asks for user name and password.
2. System verifies the name and password in databases.
3. After successful verification
   - Login shell is started with the user identity
     - real-user-ID: $RUID = UID$
     - real-group-ID: $RGID = GID_1$
     - effective-user-ID: $EUID = UID$
     - effective-group-ID: $EGID = GID_1$
     - list of group-ID's: GID_1, GID_2, ... , GID_n
   - Working directory is set to home directory of the user account

Process identity change

- Process identity is set by kernel during process startup or kernel can change it on demand of process.
- Real and effective identities of process (`RUID`, `RGID`, `EUID` and `EGID`) are the same in most cases and they are inherited from the parent process.
- In some cases, they are not inherited from the parent process and can be different
  - During login (e.g. process `systemd-logind`/`dtlogin`).
  - By commands `su`/`newgrp`.
  - Execution of binary files with special access permission `set-user-ID`.
  - Execution of binary files with specila access permission `set-group-ID`.

Classic model of security access to data

- **Process**
  - Real process identity *RUID* and *RGID*
    - It equals to the identity of the user that starts the process.
    - It defines the owner of the process.
    - It can be printed by commands

```
ps -o ruser,ruid,rgroup,rgid,comm
```

```
pcred $$                                    # only Solaris
```

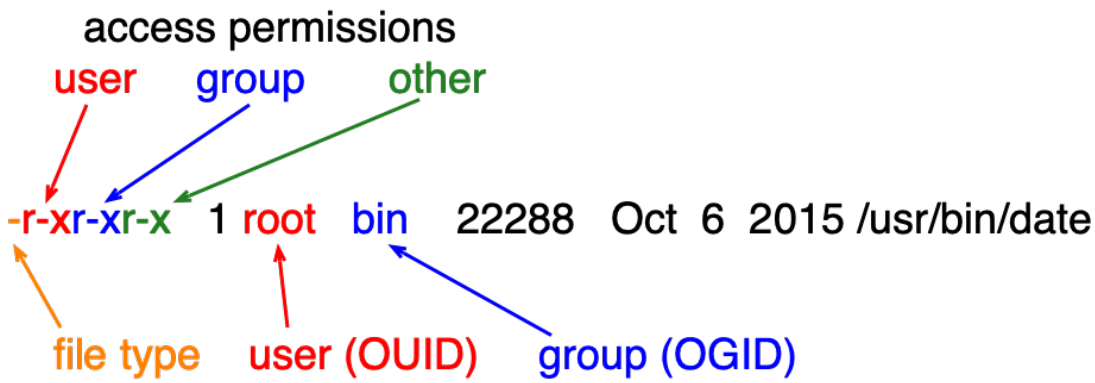
  - Effective process identity *EUID* and *EGID*
    - It is equals to real identity (by most cases).
    - It is used to verify process access to objects (files, dirctories, ...).
    - It can be printed by commands

```
ps -o user,uid,group,gid,comm
```

```
pcred $$                                    # only Solaris
```

Classic model of security access to data

- **Object (file, directory, ...)**
  - User and group identity of the owner *OUID*/*OGID*.
  - Access rights for each category
    - user,
    - group,
    - other.
  - Types of access rights
    - Basic: read, write, execute.
    - Special: set-user-ID bit, set-group-ID bit, sticky bit.
  - These information can be get by command `ls -l`.



Classic model of security access to data

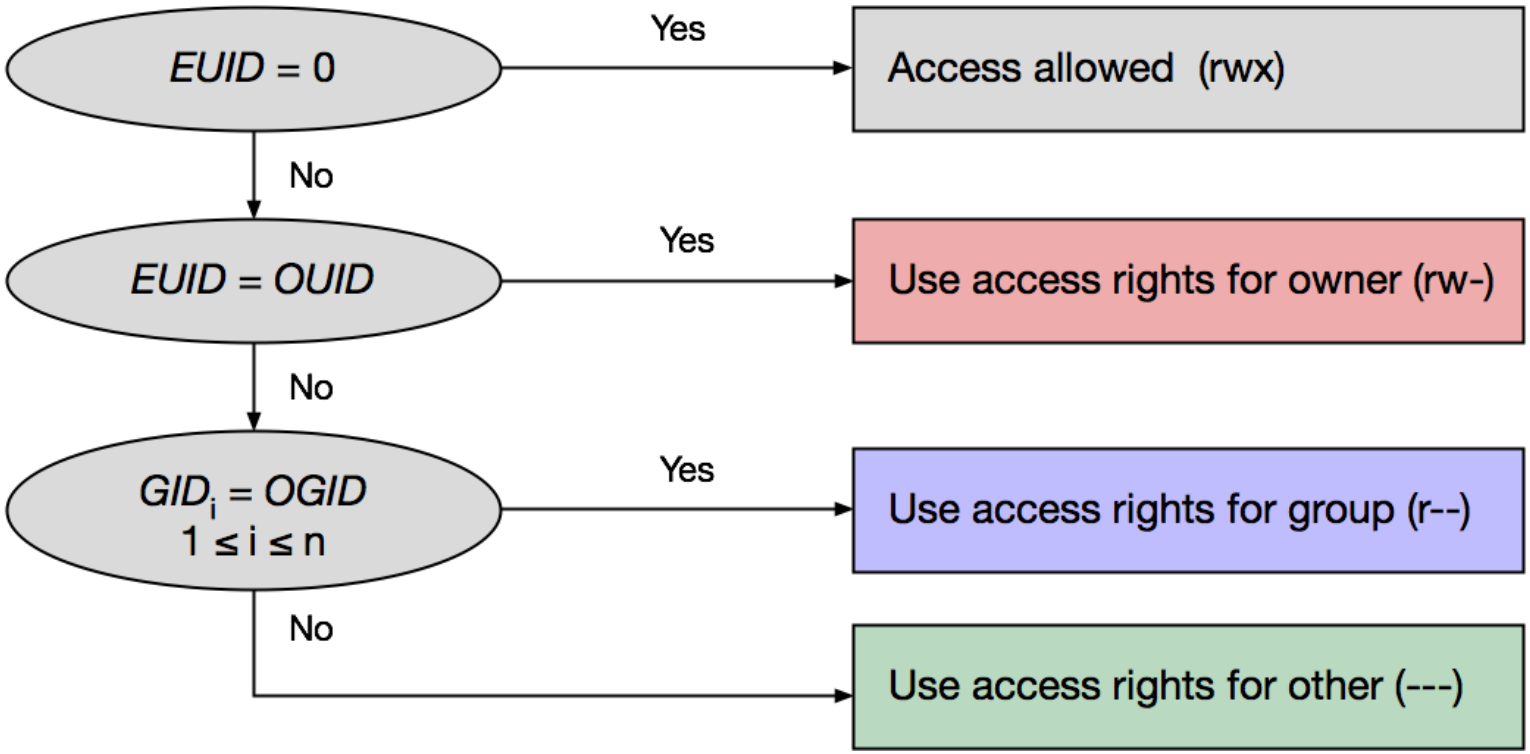
- **Access permissions**

Permission	File	Directory
r (read)	Read content of file (`cat` *file*).	List content of directory without attributes (`ls` *directory*).
w (write)	Modify content of file (`vi` *file*).	Create or remove files/subdirectory (`rm` *file*).
x (execute)	Execute program (`./`*file*).	Set and browse directory (`cd` *directory* or `ls -l` *file*).
s (set-user-ID)	After execution of binary file, the process *EUID* = *OUID*.	No meaning.
s (set-group-ID)	After execution of binary file, the process *EGID* = *OGID*.	New file inherits *OGID* form directory, not from *EGID* of process.
t (sticky bit)	No meaning.	Anybody can create item in this directory. Only owner can remove them.

Classic model of security access to data

- **Access control of processes to object**

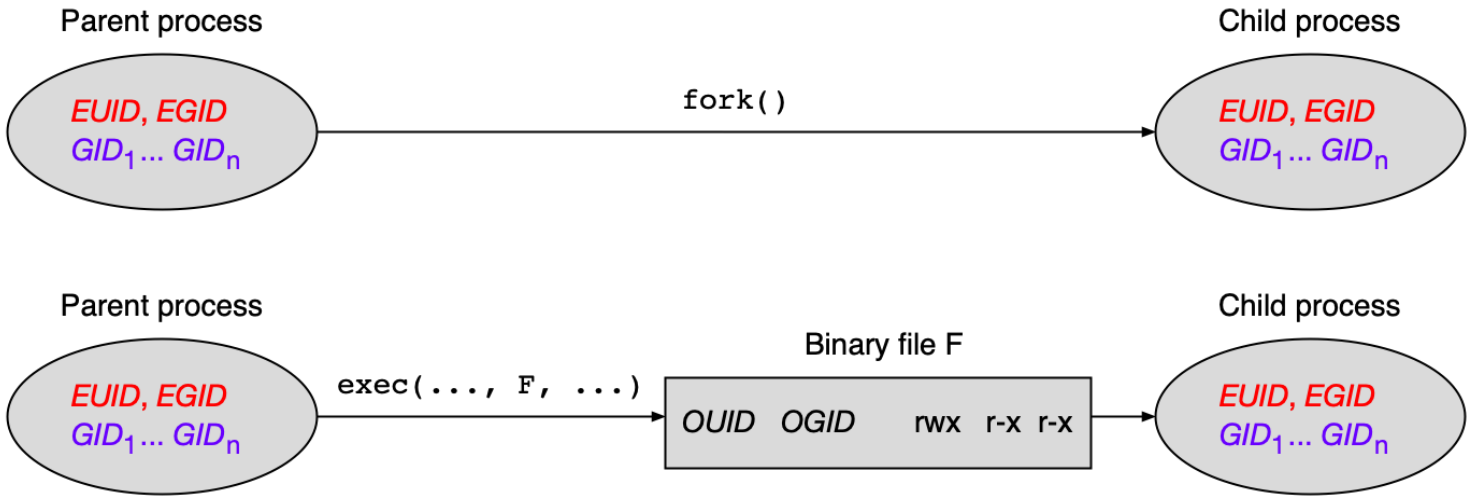
Process: *EUID*, GID_1 ... GID_n **Object:** *OUID* *OGID* rw- r-- ---



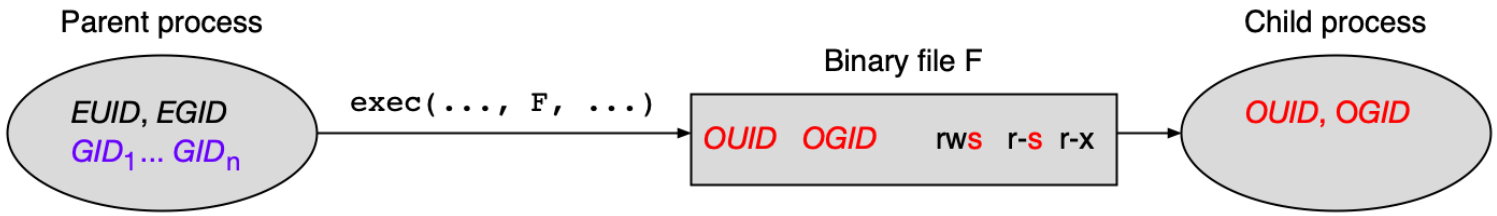
Classic model of security access to data

New process creation

- New process can be created by system call `fork()` or `exec()`.
- The identity of the process is inherited from the parent



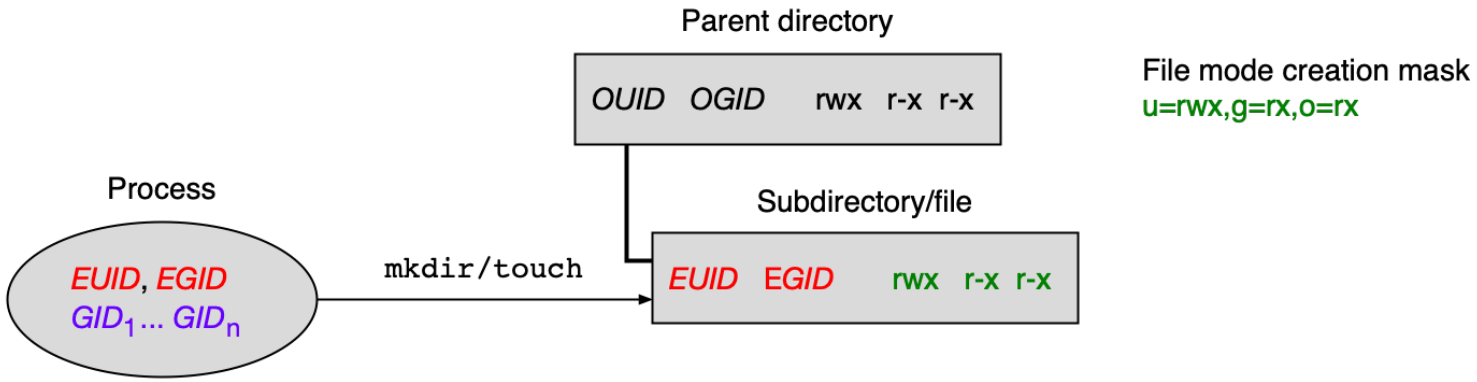
- The process identity is set according to the file owner



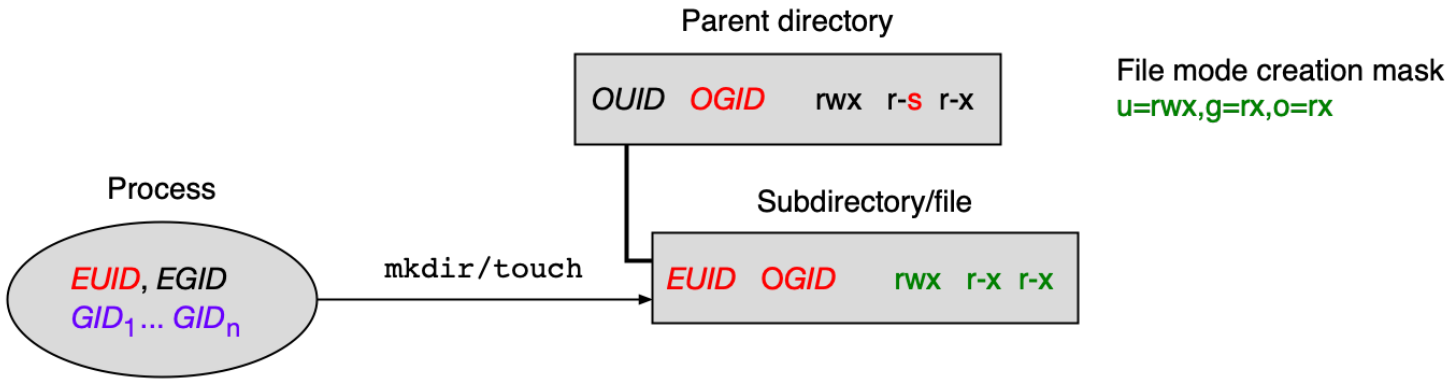
Classic model of security access to data

New file creation

- The user/group of file is inherited from the process



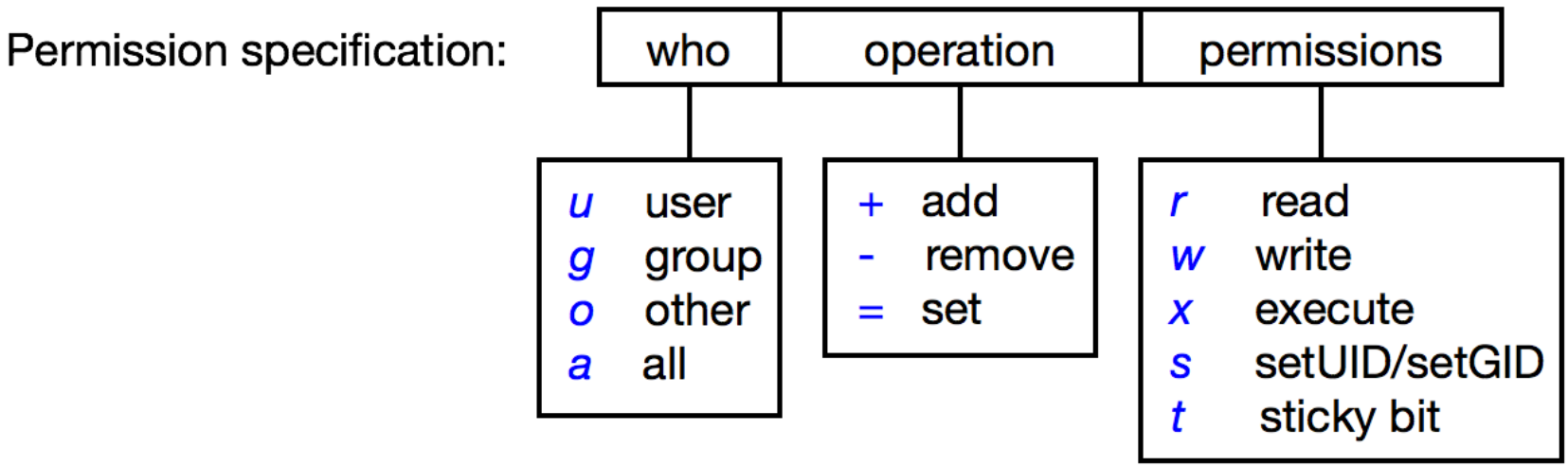
- The group of file is inherited from the directory



Ownership and access permissions settings

- User/group ownership settings
  `chown [-R] user [:group] file/directory`
  `chgrp [-R] group file/directory`
  - Only system administrator can change the ownership of file/directory.
  - Useful options
    - `-R` ... change files and directories recursively.
- Access permission settings
  `chmod [-R] symbolic_mode file/directory`
  `chmod [-R] octal_mode file/directory`
  - Useful options
    - `-R` ... change files and directories recursively.

Access permission settings – symbolic mode



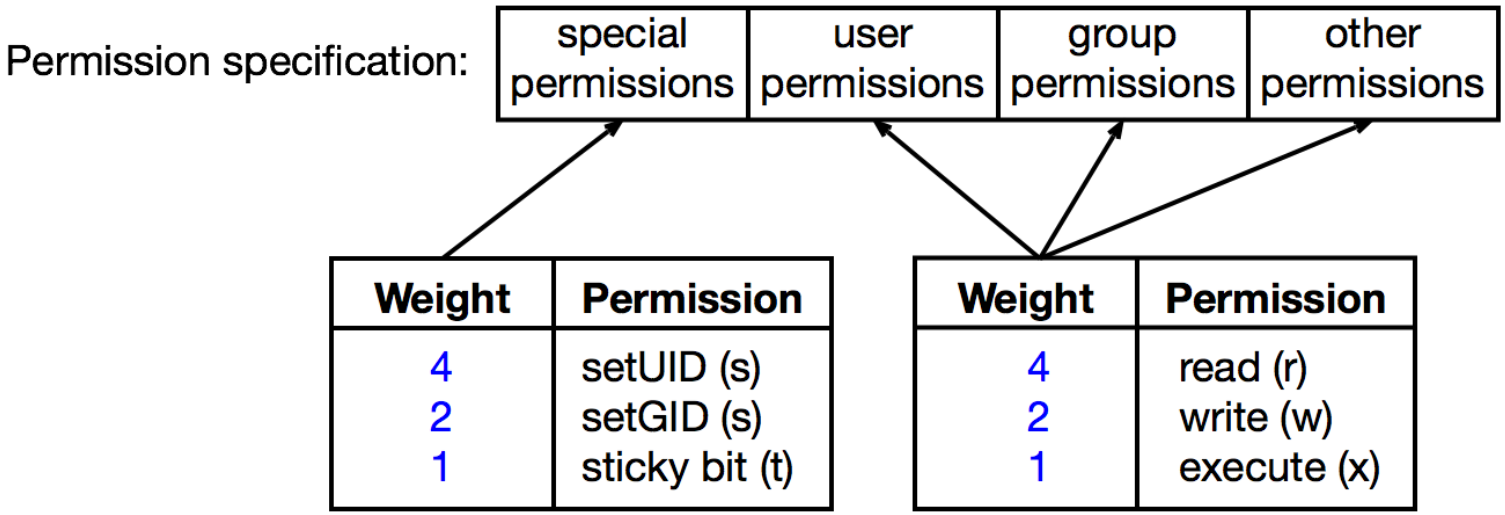
- Examples

```
~> ls -l a.txt
-rw-r--r-- 1 honza users 12 Nov 26 16:21 a.txt
```

```
~> chmod u+x,g-r,o+w a.txt
~> ls -l a.txt
-rwx---rw- 1 honza users 12 Nov 26 16:21 a.txt
```

```
~> chmod a=rx a.txt
~> ls -l a.txt
-r-xr-xr-x 1 honza users 12 Nov 26 16:21 a.txt
```

Access permission settings – absolute (octal) mode



- Examples

```
~> ls -l a.txt
-rw-r--r-- 1 honza users 12 Nov 26 16:21 a.txt
```

```
~> chmod 706 a.txt
~> ls -l a.txt
-rwx---rw- 1 honza users 12 Nov 26 16:21 a.txt
```

```
~> chmod 555 a.txt
~> ls -l a.txt
-r-xr-xr-x 1 honza users 12 Nov 26 16:21 a.txt
```

Default access permissions of new file/new directory

- The user file-creation mask defines permissions of new file/directory.
- It is inherited from the parent process.
- Maximum allowed default permissions
  - 0777 (rwx) for a new directory.
  - 0666 (rw-) for a new file.
- It can be printed or modified by command `umask`.

- Examples

```
~> umask
0022                          # forbidden rights for directory
~> umask -S
u=rwx,g=rx,o=rx               # allowed rights
```

```
~> mkdir dir ; touch file
~> ls -ld dir file
drwxr-xr-x 1 honza users 0 Nov 26 17:11 dir
-rw-r--r-- 1 honza users 0 Nov 26 17:11 file
```