Lecture 6

Process identity. Access permissions.

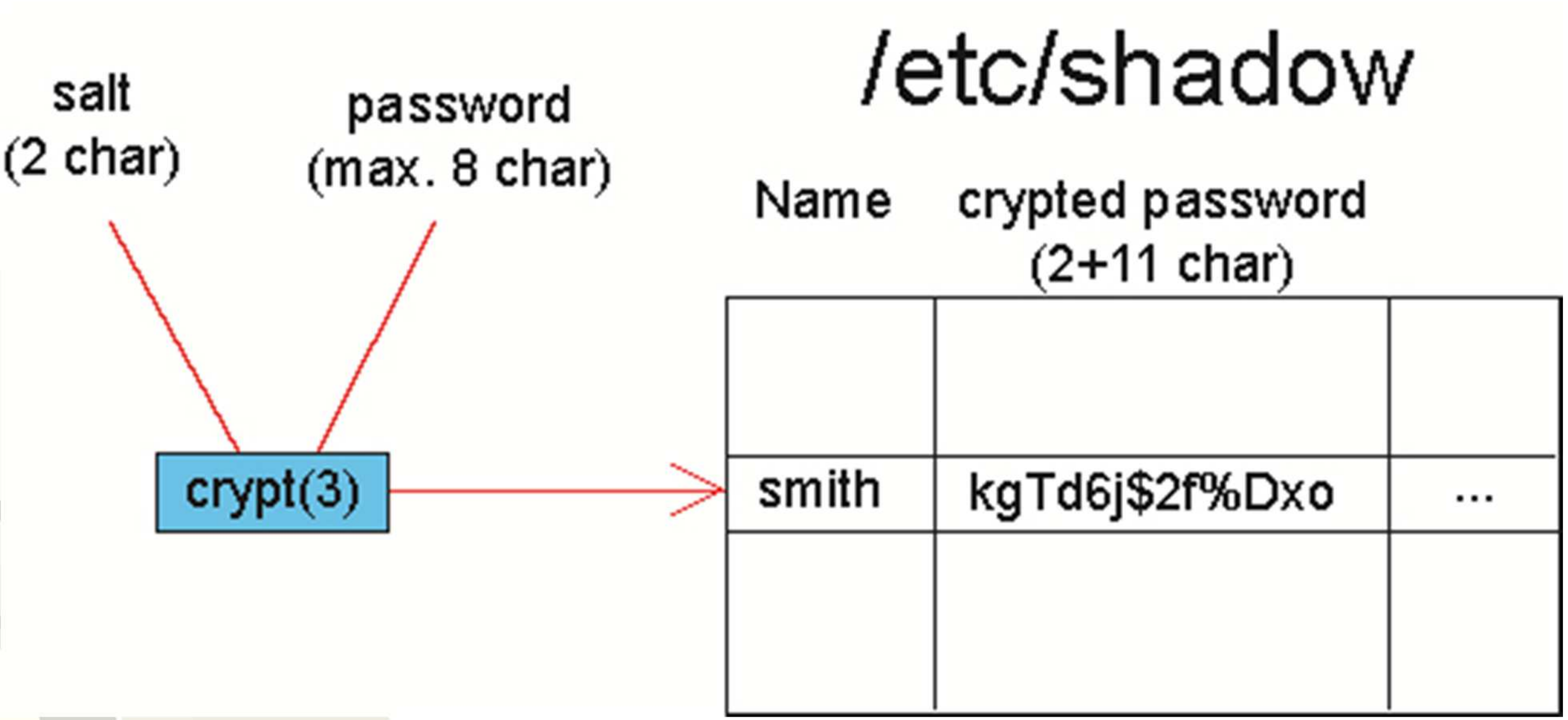
Department of Computer Systems FIT, Czech Technical University in Prague

User account

- **User name**
- **User ID (UID)**
  - UID=0 represents root account
- **Primary group ID (GID)**
  - User belongs to at least one group (primary)
- **List of secondary group ID's (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 account

- **Encrypted password**
  - It can be changed by command **`passwd`**
  - User must known the original password, root needn’t
  - User must satisfy password policy, root needn’t



Name	crypted password (2+11 char)	
smith	kgTd6j$2f%Dxo	...

User account database

- **/etc/nsswitch.conf**
  - Defines where the information about user accounts are saved

- **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

- File /etc/passwd

```
name:x:UID:GID:comment:home_directory:login_shell
```

- File /etc/shadow

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

- File /etc/group

```
group::GID:list_of_users
```

Login process

- **System asks** for user **name** and **password**
- **System verifies the name and password** in database
- **After successful verification, login shell is started** and
  - **working directory** = home directory of the user account
  - **effective user ID (EUID)** = **UID**
  - **real user ID (RUID)** = **UID**
  - **save user ID (SUID)** = **UID**
  - **effective group ID (EGID)** = primary **GID**
  - **real group ID (RGID)** = primary **GID**
  - **save group ID (SGID)** = primary **GID**
  - **list of secondary group ID's**

Process identity

- **Real process identity (RUID, RGID)**
  - It equals to the identity of the user that starts the process (by default)
  - Can be printed by commands

```
ps -o ruid,rgid,comm
pcred PID
```

- **Saved process identity (RUID, RGID)**
  - It is equals to real identity (in most cases)
  - It can be printed by commands

```
pcred PID
```

- **Effective process identity (EUID, EGID)**
  - It is used to **verify the authenticity** of the process
  - It is equals to real identity (in most cases)
  - It can be changed to real or saved identity
  - It can be printed by commands

```
ps –o uid,gid,comm
pcred PID
```

Modification of process identity

- **Process identity is set by kernel** during process startup or kernel can change it on demand of process.

- **`RUID`**, **`EUID`**, **`SUID`** respectively **`RGID`**, **`EGID`**, **`SGID`** are the same in most cases and they are **inherited from the parent process**.

- **In some cases**, they are not inherited from parent and can be different:
  - During login (process **`login`**/**`dtlogin`**)
  - By commands **`su`**/**`newgrp`**
  - Execution of binary files with permission **`suid`** modifies **`EUID,SUID`**
  - Execution of binary files with permission **`guid`** modifies **`EGID,SGID`**

su / newgrp

```
su [ - ] [ user_name]
```

- Stars new login shell with new process identity
- With option – new environment is set

```
newgrp secondary_group
```

- Start new shell with group identity=secondary group

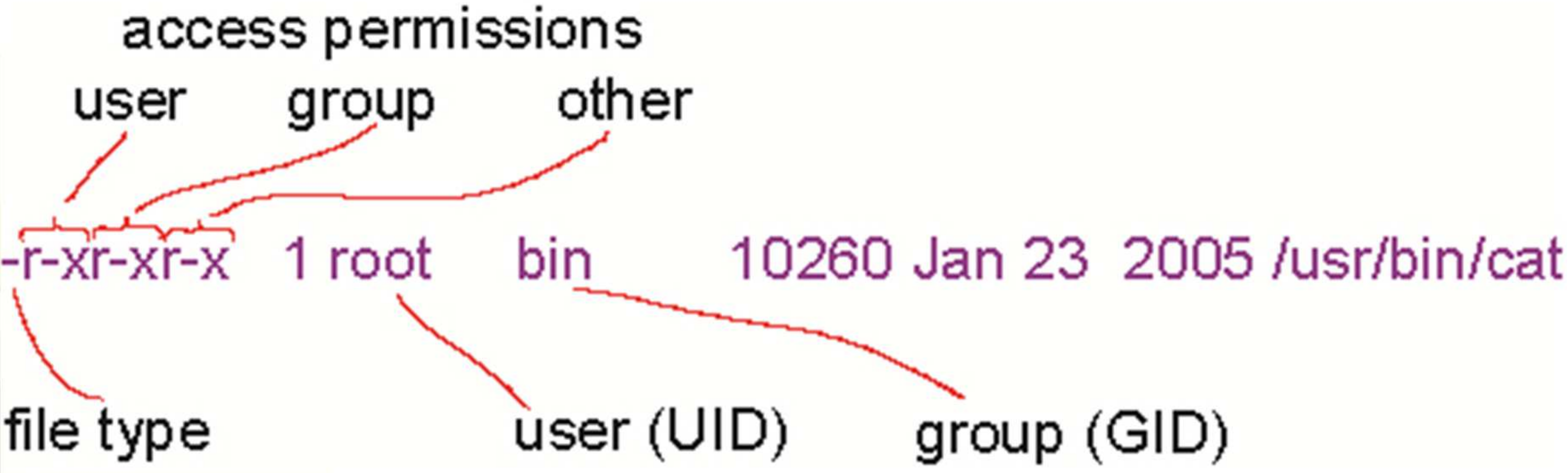
Example

```
$ id
uid=0(root) gid=1(other)
$ su - trdlicka
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
You have new mail.
$ id -a
uid=4365(trdlicka) gid=1002(k336) groups=1002(k336),2003(y36uos)
$ newgrp y36uos
$ id
uid=4365(trdlicka) gid=2003(y36uos)
$ newgrp k336
$ id
uid=4365(trdlicka) gid=1002(k336)
```

Access permissions

- **File/directory**
  - Owner-user (**`UID`**)
  - Owner-group (**`GID`**)
  - Access permissions **r**ead, **w**rite and e**x**ecute for **u**ser, **g**roup and **o**ther.

- These information about file can be printed by command: **`ls –l`**

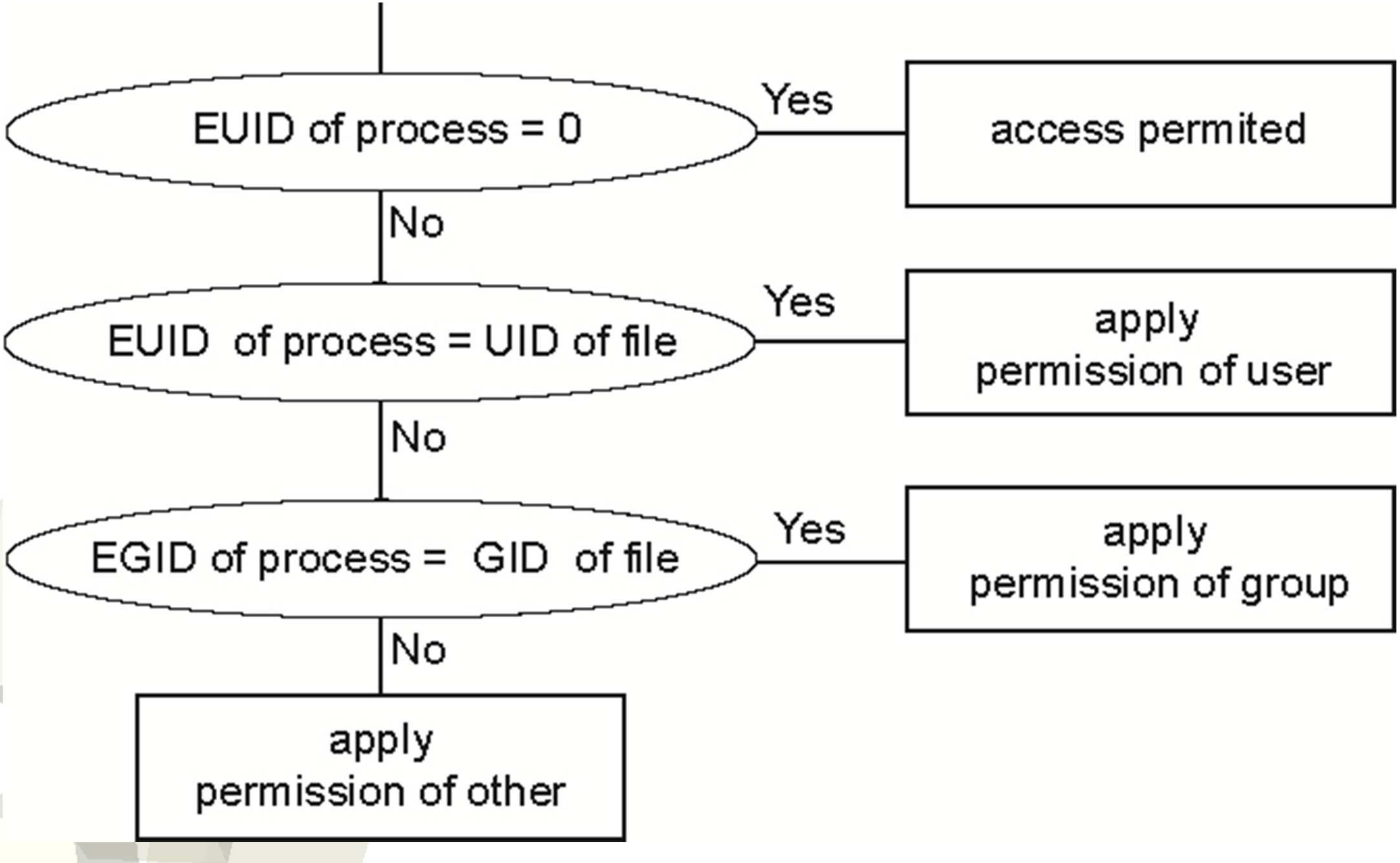


```
-r-xr-xr-x  1 root   bin    10260 Jan 23  2005 /usr/bin/cat
```

Access permissions

Permissions	File	Directory
r	Read content of file (`cat`)	List content of directory (`ls`) without atributes
w	Modify content of file (`vi`)	Create or remove files/subdirectory (`rm`)
x	Execute program	set and browse directory (`cd`)

Access permissions



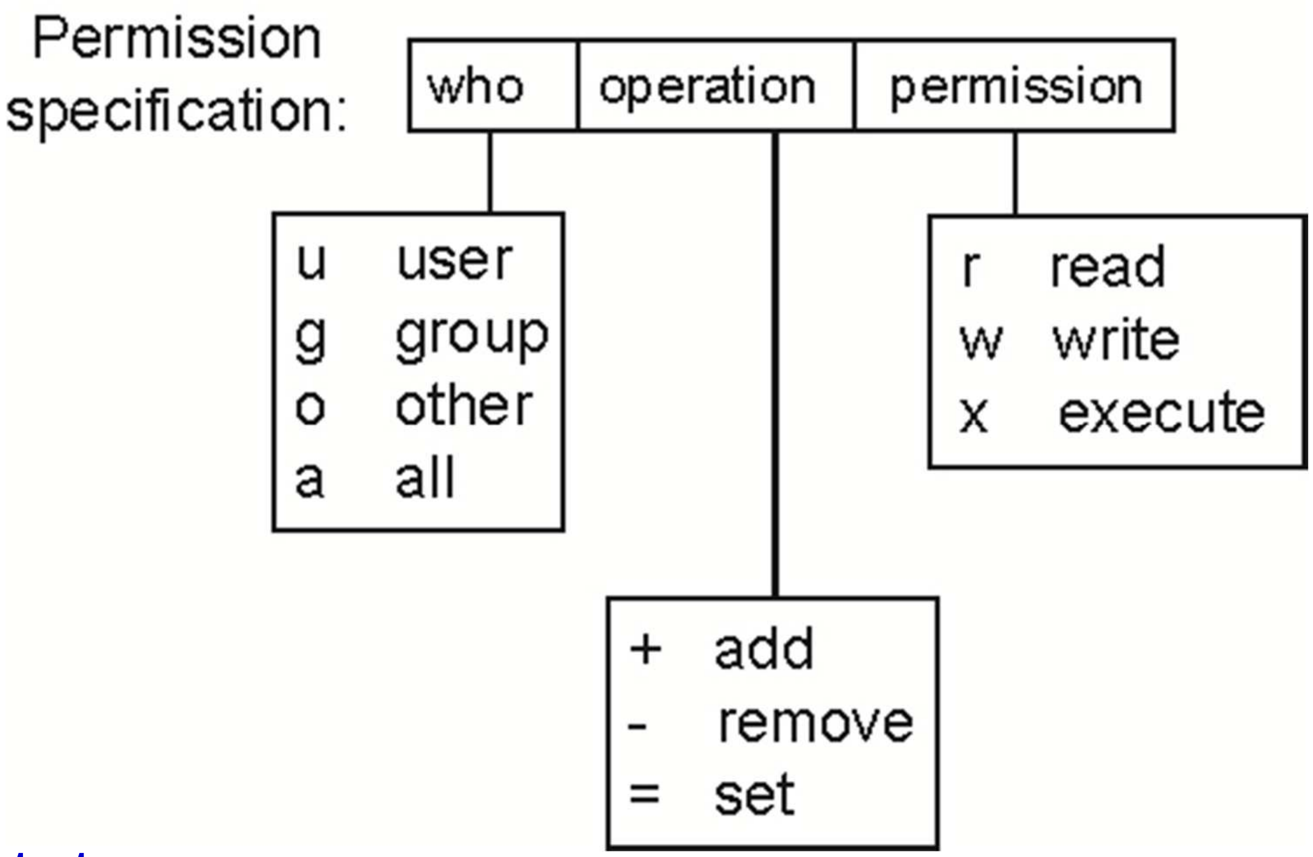
Access permissions

```
chmod [-R] permissions files
```

-R Recursively descends through directory arguments

`permissions` can be defined by symbolic or absolute (octal) mode

Symbolic mode



Permission specification:

who | operation | permission

u user
g group
o other
a all

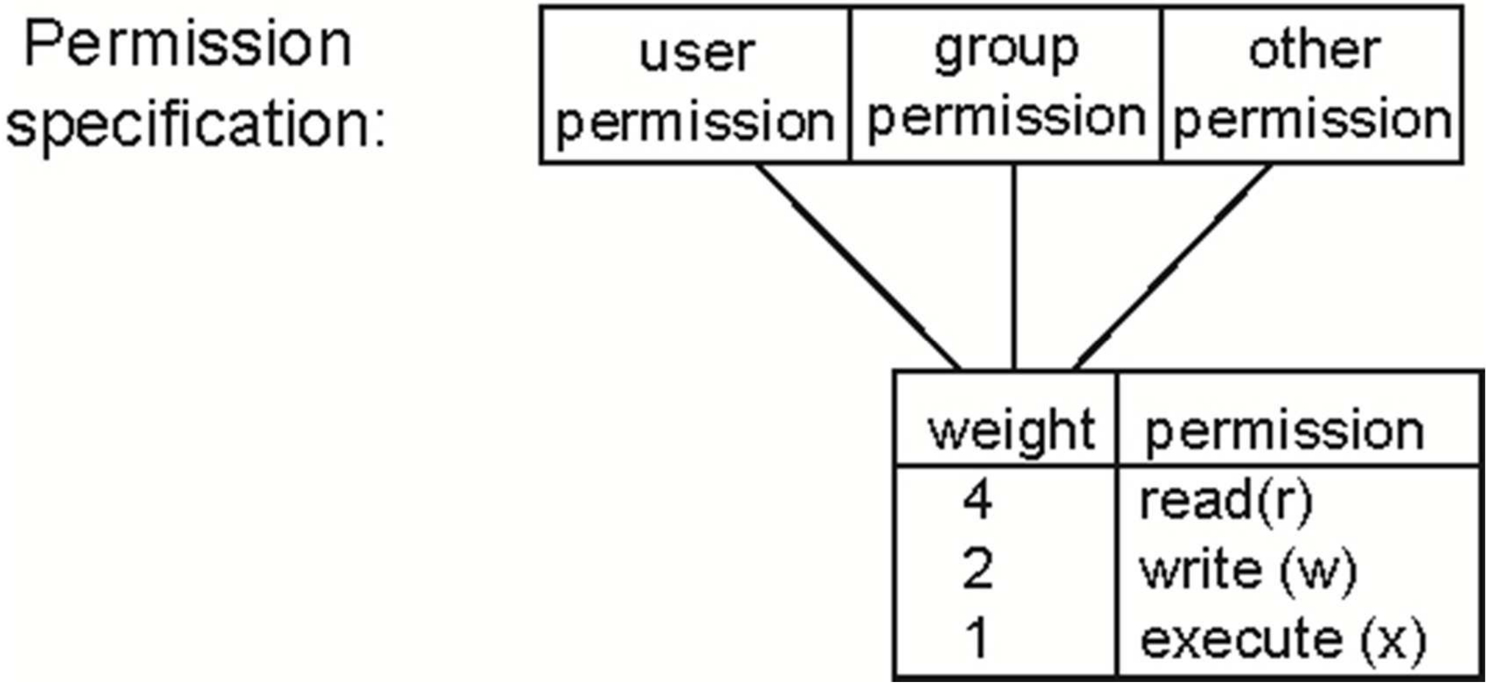
r read
w write
x execute

+ add
- remove
= set

Example:

```
$ ls -l a.txt
-rw-r--r--   1 trdlicka k336       1105 Oct 23 20:52 a.txt
$ chmod u+x,g-r,o+w a.txt
$ ls -l a.txt
-rwx---rw-   1 trdlicka k336       1105 Oct 23 20:52 a.txt
```

Absolute (octal) mode



Permission specification:

user permission	group permission	other permission

weight	permission
4	read(r)
2	write (w)
1	execute (x)

Example:

```
$ ls -l a.txt
-rw-r--r--   1 trdlicka k336       1105 Oct 23 20:52 a.txt
$ chmod 706 a.txt
$ ls -l a.txt
-rwx---rw-   1 trdlicka k336       1105 Oct 23 20:52 a.txt
```

File mode creation mask

- It defines permissions that are set to a file/directory during its creation.
- It is inherited from parent process.
- It can be printed or modified by command `umask` .
- File permissions = initial permissions ∩ file mode creation mask
- **File Initial permissions are 666**
- **Directory initial permissions are 777**

mask	file	directory	note
000	666	777	Initial value, unsafe
022	644	755	Default
027	640	750	More safe
077	600	700	The most safe
066	600	711	Compromise

How to change user/group owner

- **Only root can.**
- By commands

```
chown [-R] user [:group] files
chgrp [-R] group files
```

Special access permissions

Permissions	Value	File	Directory
suid	4000 u+s	After execution of binary file, the process EUID equals to owner of binary file	No meaning
sgid **l**ock	2000 g+s	After execution of binary file, the process EGID equals to group of binary file	New files from directory inherit GID of directory, not of GID of process
s**t**icky s**T**icky	1000 o+t	No meaning	Anybody can create file/directory in this directory. Only owner can remove them.