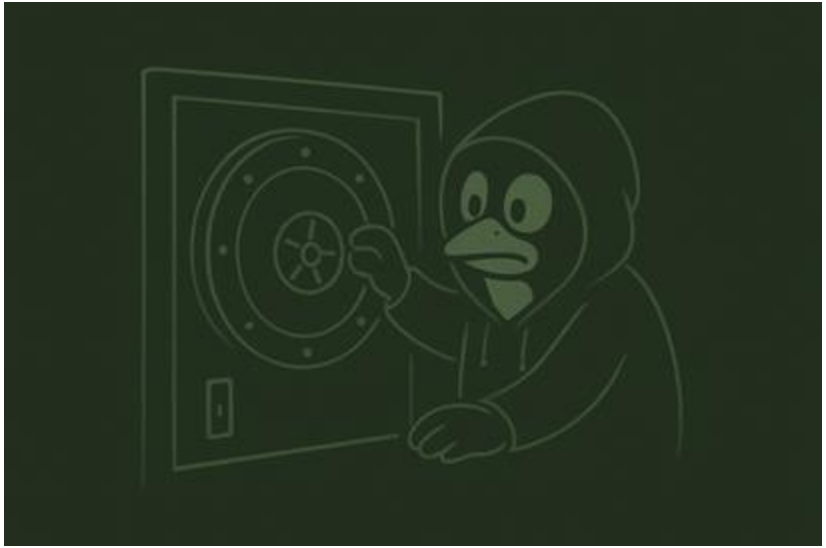
Operating System fundamentals

Access control



Karel de Grote Hogeschool

Contents

1. Understanding Linux permissions
2. Setting Linux permissions
3. Special and default permissions

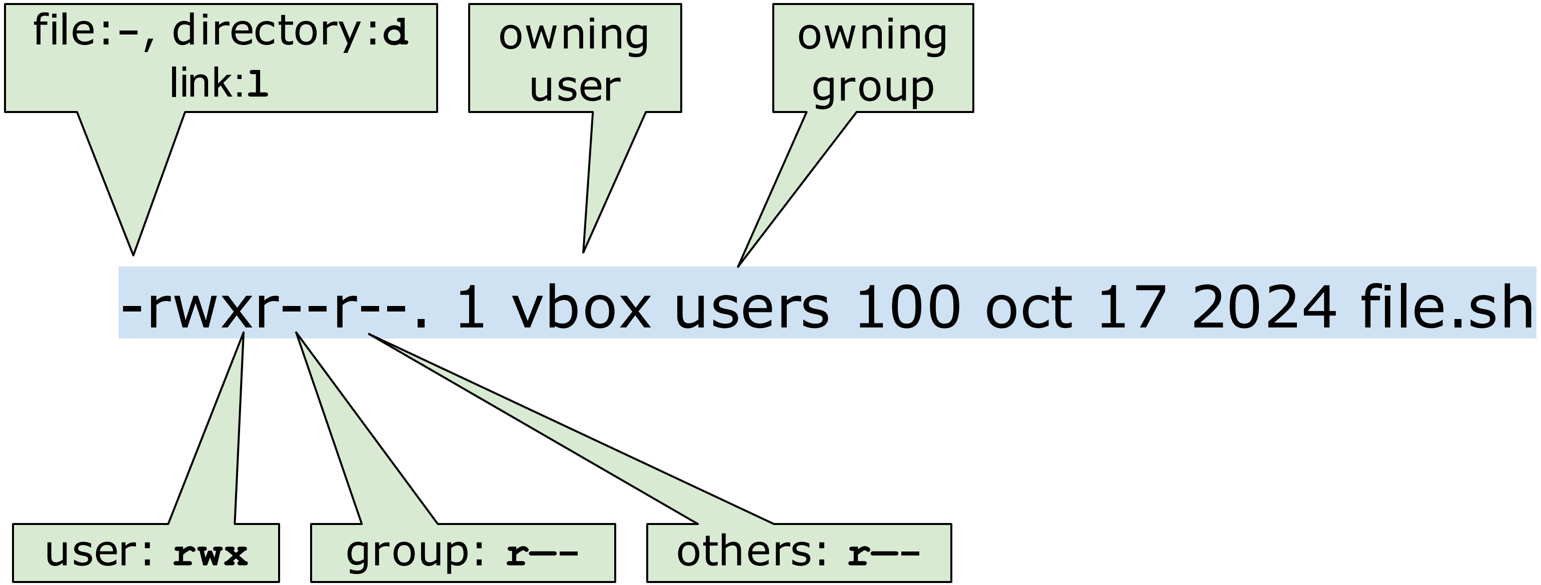
Course text

- Chapter 11
  - (RedHat chapter 7)
  - Interpret Linux File System Permissions
  - Manage File System Permissions from the Command Line
  - Manage Default Permissions and File Access

Understanding Linux permissions

1

Files belong to a user and a group: ls -l



file:–, directory:**`d`**
link:**`l`**

owning user

owning group

```
-rwxr--r--. 1 vbox users 100 oct 17 2024 file.sh
```

user: **`rwx`**

group: **`r——`**

others: **`r——`**

Permissions

	file	directory
r READ	open en read	get list of files and directories
w WRITE	change contents	create or delete files and directories
x EXECUTE	execute binary file or script	access to the directory (with cd) (used together with r)

Exercise

- RedHat
  - ch07s02
- explain ownership and permissions of:
  - /etc/shadow
  - ~/.bashrc
  - ~/.bash_history
  - /var/log

Setting Linux permissions

2

Changing owning user en group

- You can set the owning user with:

  chown username file

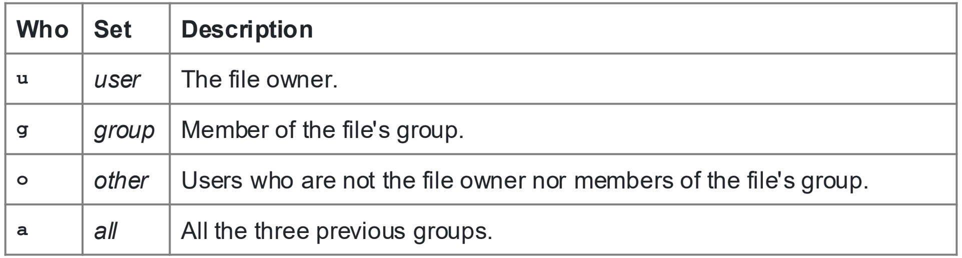
- You can set the owning group with:

  chgrp groupname file

  chown :groupname file

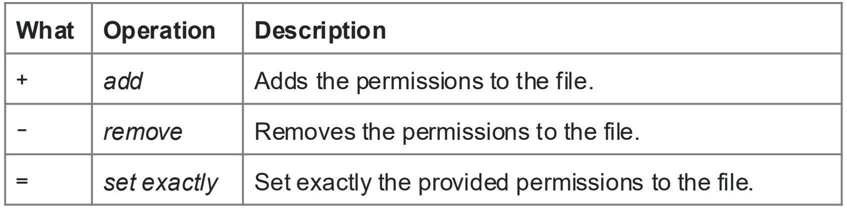
- You can change them both in one command:
  - **chown** username:groupname file

Setting permissions



Who	Set	Description
`u`	*user*	The file owner.
`g`	*group*	Member of the file's group.
`o`	*other*	Users who are not the file owner nor members of the file's group.
`a`	*all*	All the three previous groups.

- **chmod**
  - **chmod** +x file -> add x at all three places
  - **chmod** u+x file -> add x for user
  - **chmod** g+w file -> add w for group
  - **chmod** o-r file -> remove r for others
  - **chmod** g+rw,o+r file -> add r and w for group and r for others
  - **chmod** -R o-r * -> remove r for others for all files and do this recursively

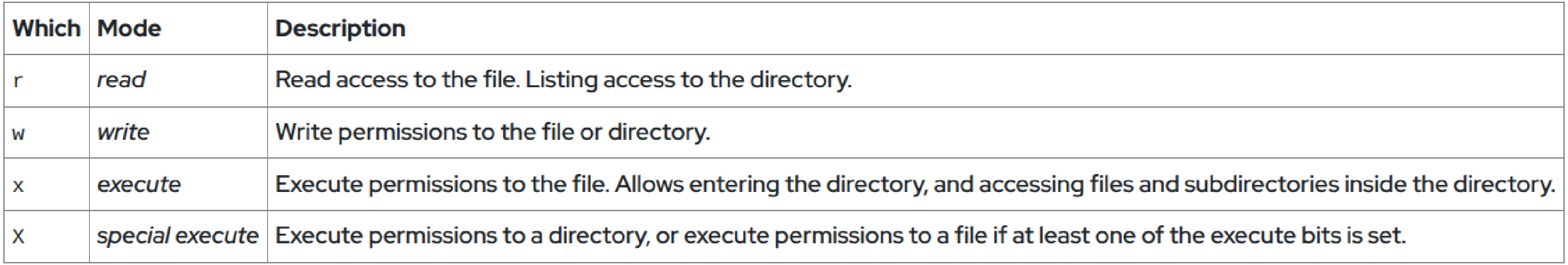


What	Operation	Description
+	*add*	Adds the permissions to the file.
-	*remove*	Removes the permissions to the file.
=	*set exactly*	Set exactly the provided permissions to the file.

Setting permissions

Who	Set	Description
`u`	*user*	The file owner.
`g`	*group*	Member of the file's group.
`o`	*other*	Users who are not the file owner nor members of the file's group.
`a`	*all*	All the three previous groups.

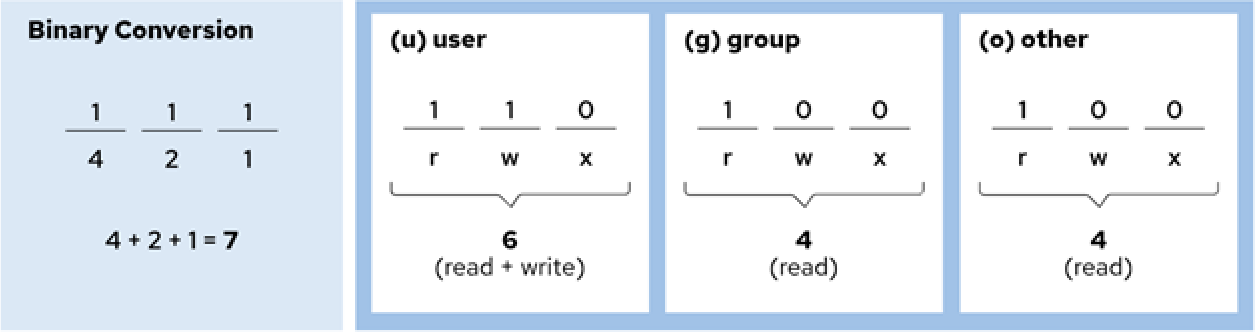
What	Operation	Description
+	*add*	Adds the permissions to the file.
-	*remove*	Removes the permissions to the file.
=	*set exactly*	Set exactly the provided permissions to the file.



Which	Mode	Description
`r`	*read*	Read access to the file. Listing access to the directory.
`w`	*write*	Write permissions to the file or directory.
`x`	*execute*	Execute permissions to the file. Allows entering the directory, and accessing files and subdirectories inside the directory.
`X`	*special execute*	Execute permissions to a directory, or execute permissions to a file if at least one of the execute bits is set.

Setting permissions

- **chmod** octal
  - use an octal number for the permissions
  - r = 4, w = 2, x = 1
- examples:
  - **chmod** 755 directory **`drwxr-xr-x`**
  - **chmod** 640 file **`-rw-r-----`**
  - **chmod** 644 file **`-rw-r--r--`**



Exercise

```
cd ~
mkdir t
ls –l

chmod 400 t
ls -l
ls t
touch t/test

chmod 500 t
ls -l
ls t
touch t/test

chmod 700 t
ls -l
ls t
touch t/test
```

Special and default permissions

Karel de Grote Hogeschool

Special permissions

- problems:
  - a program always executes with the permissions of the user that initiated it -> what if you want to change your password?
  - a mail daemon runs under its own account and saves incoming emails in a folder. The receiving user needs access
  - in a common directory you can also delete files belonging to somebody else…

Special permissions

- take a look at the permissions of:
  - ls -l /usr/bin/passwd
  - ls -l /usr/bin/locate (not on headless)
  - ls -ld /tmp

Special permissions

	file	directory
SUID	execute the file with the permissions of the owner (*does not work with scripts)	no meaning
SGID	execute the file with the permissions of the group (*does not work with scripts)	everything created in this directory will be owned by the owning group of the directory
Sticky bit	no meaning	only the owner of a file can delete it

*Does not work with scripts because the system reads these as text files.

Setting special permissions

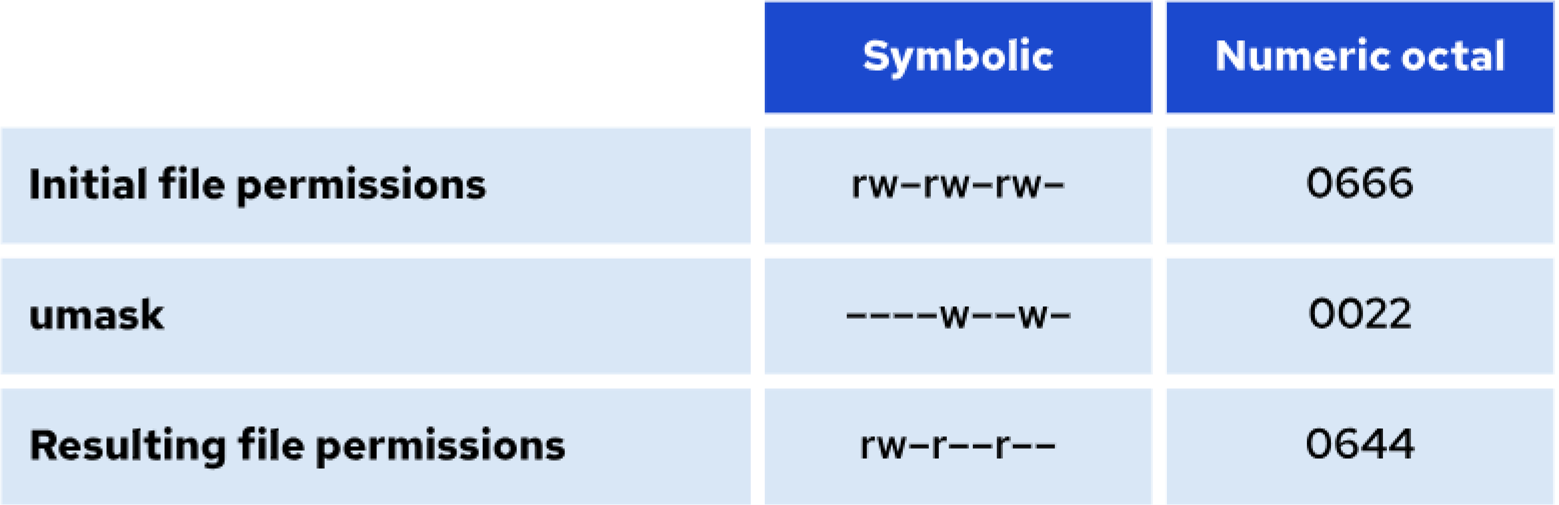
- 4th octal number:
  - **chmod 4777** program -> **-rwsrwxrwx**
  - **chmod 2777** map -> **drwxrwsrwx**
  - **chmod 1777** map -> **drwxrwxrwt**
- with named parameters:
  - **chmod u+s** program
  - **chmod g+s** directory/program
  - **chmod +t** directory

Default permissions

- What permissions are assigned to new files/directories?
  - touch text.txt
  - mkdir new_dir
- This can be changed using “**umask**”
- umask contains a “mask” that removes default permissions
- e.g.:
  - umask -> shows current value (0022)
  - umask 0000 -> all access for everyone
  - umask 0077 -> all access for owner, nothing for the rest

Default permissions using umask

- Default file 0666
  Default folder 0777



	Symbolic	Numeric octal
Initial file permissions	rw-rw-rw-	0666
umask	----w--w-	0022
Resulting file permissions	rw-r--r--	0644

4

Exercises

Exercises

- KdG
  - 11.1 till 11.8
- RedHat
  - ch07s02
  - ch07s04
  - ch07s06
  - ch07s07

KdG Karel de Grote
Hogeschool