

“Київський фаховий коледж зв’язку”

Циклова комісія Комп’ютерної інженерії

ЗВІТ ПО ВИКОНАННЮ ЛАБОРАТОРНОЇ РОБОТИ №10

з дисципліни: «Операційні системи»

**Тема: «Зміна власників і прав доступу до
файлів в Linux. Спеціальні каталоги
та файли в Linux»**

Виконала студентка
групи РПЗ-136
Дімітрова С.П.
Перевірів викладач
Сушанова В.С.

Київ 2024

Мета роботи:

1. Отримання практичних навиків роботи з командною оболонкою Bash.
2. Знайомство з базовими діями при зміні власників файлів, прав доступу до файлів
3. Знайомство з спеціальними каталогами та файлами в Linux.

Матеріальне забезпечення занять:

1. ЕОМ типу IBM PC.
2. ОС сімейства Windows та віртуальна машина Virtual Box (Oracle).
3. ОС GNU/Linux (будь-який дистрибутив).
4. Сайт мережевої академії Cisco netacad.com та його онлайн курси по Linux

Завдання для попередньої підготовки:

- *Прочитайте короткі теоретичні відомості до лабораторної роботи та зробіть невеликий словник базових англійських термінів з питань призначення команд та їх параметрів.

| Термін англійською | Термін українською |
|-----------------------|---|
| File ownership | Право власності на файл. Вказує на користувача та групу, які мають право доступу до файлу. |
| Supplemental groups | Додаткові групи |
| Setuid (Set User ID) | Установка ідентифікатора користувача. Якщо setuid встановлено, то при запуску файлу, процес отримує ідентифікатор користувача файлу, а не ідентифікатор користувача, яким запущений процес. |
| Setgid (Set Group ID) | Установка ідентифікатора групи. Якщо setgid встановлено для файлу, то при запуску процесу, процес отримує ідентифікатор групи файлу, а не ідентифікатор групи, до якої належить користувач. |
| Sticky bit | Якщо sticky bit встановлено для директорії, тоді тільки власник файлу може його видалити або перейменувати. Цей біт використовується, наприклад, для захисту спільних директорій, де кілька користувачів можуть записувати файли. |

| | |
|-----------------|--|
| Octal method | Вісімковий метод. Метод представлення прав доступу до файлу за допомогою вісімкових чисел. . Кожні три біти відповідають окремій категорії користувачів (власник, група та інші користувачі), а значення кожного біта відображає наявність або відсутність певних прав доступу (читання, запису та виконання). |
| Symbolic method | Символьний метод. Метод представлення прав доступу до файлу за допомогою символів. Кожна категорія користувачів (власник, група та ін. користувачі) представлена символами (r - читання, w - запис, x - виконання), які вказують на наявність або відсутність певних прав доступу. |

- Вивчіть матеріали онлайн-курсу академії Cisco “NDG Linux Essentials”:
- Chapter 17 - Ownership and Permissions
- Chapter 18 - Special Directories and Files
- Пройдіть тестування у курсі NDG Linux Essentials за такими темами:
- Chapter 17 Exam

Home / I'm Learning / KFKZ_Linux Essentials_2024(RPZ+BIKS) / Module 17 - Ownership and Permissions / Chapter 17 Exam

KFKZ_Linux Essentials_2024(RPZ+BIKS)

Chapter 17 Exam

| Previous Attempts | | | |
|-----------------------------|------|---------|----------------------|
| 1 attempts with 2 remaining | | | |
| April 13, 2024, 3:13 a.m. | 100% | 10 / 10 | View |

Warning
You received a grade of 100% on your last attempt. If you take this assessment again your new grade will replace your previous grade.

You have taken this assessment 1 times. You have 2 attempts remaining. Click the Begin button to get started. Make sure you complete all questions and submit your answers when you are done.

[Begin Assessment](#)

- Chapter 18 Exam

Home / I'm Learning / KFKZ_Linux Essentials_2024(RPZ+BIKS) / Module 18 - Special Directories and Files / Chapter 18 Exam

KFKZ_Linux Essentials_2024(RPZ+BIKS)

Chapter 18 Exam

| Previous Attempts | | | |
|-----------------------------|------|---------|----------------------|
| 1 attempts with 2 remaining | | | |
| April 13, 2024, 12:56 p.m. | 100% | 10 / 10 | View |

Warning
You received a grade of 100% on your last attempt. If you take this assessment again your new grade will replace your previous grade.

You have taken this assessment 1 times. You have 2 attempts remaining. Click the Begin button to get started. Make sure you complete all questions and submit your answers when you are done.

[Begin Assessment](#)

- На базі розглянутого матеріалу дайте відповіді на наступні питання:

4.1 Яке призначення команди `id`?

The `id` command is used to display information about the current user and their groups. This includes the user ID (UID), username, group ID (GID), primary group name, and any supplemental groups the user belongs to. It allows to check which user account is currently in use and which groups are available for use.



```
sofipxs@ubuntu:~$ id
uid=1000(sofipxs) gid=1000(sofipxs) groups=1000(sofipxs),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd)
sofipxs@ubuntu:~$
```

4.2 Як переглянути які права доступу має власник файлу?

To view the access rights of the file owner, use the `ls` command with the `-l` option, which displays detailed information about files and directories in a "long listing" format.

The first character - or *d* indicates the type of file. - indicates a regular file, and *d* indicates a directory. The second three characters indicate the access rights of the file owner. The first character indicates whether the owner has read permission (r). The second character indicates whether the owner has write permission (w). The third character indicates whether the owner has the right to execute (x).



```
sofipxs@ubuntu:~$ ls -l lab8.txt
-rw-rw-r-- 1 sofipxs sofipxs 32 Apr 15 13:01 lab8.txt
sofipxs@ubuntu:~$
```

Example: `ls -l file.txt`

The output will look like the following: `-rw-rw-r-- 1 owner group 32 Apr 15 13:01 file.txt`

In the example, `-rw-rw-r--` shows the access rights of the owner of the file `lab8.txt`. Here is the decoding of these rights:

- The first character - indicates the type of file, in this case it is a regular file (not a symbolic link, not a directory, etc.).
- `rw-` means that the owner of the file (sofipxs) has read (r) and write (w) permissions, but no execute (-) permissions.
- `rw-` means that the group to which the file belongs (also sofipxs) has read (r) and write (w) permissions, but no execute (-) permissions.
- `r--` means that other users who do not belong to the group of the file owner have read (r) permission only, but no write (-) or execute (-) permission.

4.3 *Як змінити власника групи?

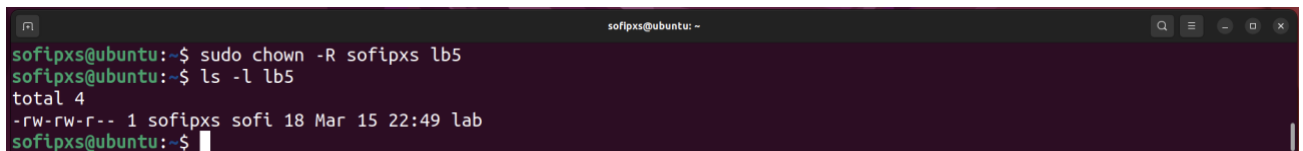
To change the group owner of a file, we can use the *chown* command (change owner). The *chown* command can only be executed by the root user and it can change both the user and group that owns a file.

If the root user wanted to change the user ownership of the *filetest1* file to the user *jane*, then the following command could be executed:

```
root@localhost:~# chown jane /tmp/filetest1
```

```
root@localhost:~# ls -l /tmp/filetest1
```

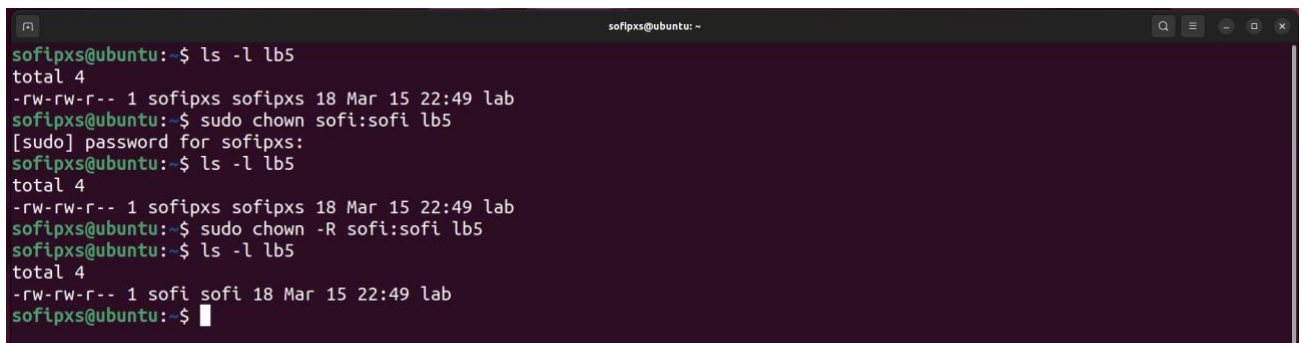
```
-rw-rw-r-- 1 jane sysadmin 0 Dec 19 18:44 /tmp/filetest1
```



```
sofipxs@ubuntu: ~  
sofipxs@ubuntu:~$ sudo chown -R sofipxs lb5  
sofipxs@ubuntu:~$ ls -l lb5  
total 4  
-rw-rw-r-- 1 sofipxs sofi 18 Mar 15 22:49 lab  
sofipxs@ubuntu:~$
```

If we want to change the group owner of the *file.txt* file to the user *new_owner* and the group *new_group*, the command will look like this:

```
sudo chown -R new_owner:new_group file.txt
```

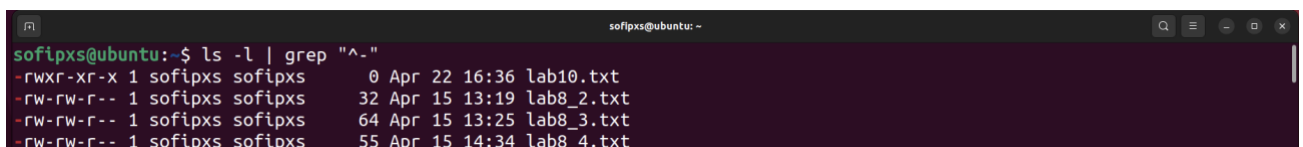


```
sofipxs@ubuntu: ~  
sofipxs@ubuntu:~$ ls -l lb5  
total 4  
-rw-rw-r-- 1 sofipxs sofipxs 18 Mar 15 22:49 lab  
sofipxs@ubuntu:~$ sudo chown sofi:sofi lb5  
[sudo] password for sofipxs:  
sofipxs@ubuntu:~$ ls -l lb5  
total 4  
-rw-rw-r-- 1 sofipxs sofipxs 18 Mar 15 22:49 lab  
sofipxs@ubuntu:~$ sudo chown -R sofi:sofi lb5  
sofipxs@ubuntu:~$ ls -l lb5  
total 4  
-rw-rw-r-- 1 sofi sofi 18 Mar 15 22:49 lab  
sofipxs@ubuntu:~$
```

4.4 *Як можна переглянути у терміналі який тип поточного файлу? Наведіть приклади для різних типів файлів.

The output of the *ls -l* command displays ten characters at the beginning of each line. These characters indicate the type of file and the permissions of the file. The first character of each line indicates the type of file. Possible values for file types:

- : A regular file, which may be empty, or contain text or binary data.



```
sofipxs@ubuntu: ~  
sofipxs@ubuntu:~$ ls -l | grep "^-"  
-rwxr-xr-x 1 sofipxs sofipxs 0 Apr 22 16:36 lab10.txt  
-rw-rw-r-- 1 sofipxs sofipxs 32 Apr 15 13:19 lab8_2.txt  
-rw-rw-r-- 1 sofipxs sofipxs 64 Apr 15 13:25 lab8_3.txt  
-rw-rw-r-- 1 sofipxs sofipxs 55 Apr 15 14:34 lab8_4.txt
```

d: A directory file, which contains the names of other files and links to them.

```
sofipxs@ubuntu:~$ ls -l | grep "^d"
drwxr-xr-x 2 sofipxs sofipxs 4096 Apr 22 16:12 Desktop
drwxr-xr-x 2 sofipxs sofipxs 4096 Apr 14 22:43 Documents
drwxr-xr-x 2 sofipxs sofipxs 4096 Mar 24 20:37 Downloads
drwxrwxr-x 2 sofipxs sofipxs 4096 Apr 22 15:37 lab10
```

l: A symbolic link is a file name that refers (points) to another file.

```
root@shantanu-Dell-System-XPS-L502X:/dev# ls -l | grep ^l
lrwxrwxrwx 1 root root 3 Jul 18 10:47 cdrom -> sr0
lrwxrwxrwx 1 root root 3 Jul 18 10:47 cdrw -> sr0
```

b: A block file is one that relates to a block hardware device where data is read in blocks of data.

```
root@shantanu-Dell-System-XPS-L502X:/dev# ls -l | grep ^b
brw-rw---- 1 root disk 7, 0 Jul 18 10:47 loop0
brw-rw---- 1 root disk 7, 1 Jul 18 10:47 loop1
brw-rw---- 1 root disk 7, 10 Jul 18 11:32 loop10
```

c: A character file is one that relates to a character hardware device where data is read one byte at a time.

```
root@shantanu-Dell-System-XPS-L502X:/dev# ls -l | grep ^c
crw-r--r-- 1 root root 10, 235 Jul 18 10:47 autofs
crw----- 1 root root 10, 234 Jul 18 10:45 btrfs-control
crw----- 1 root root 5, 1 Jul 18 10:47 console
```

p: A pipe file works similar to the pipe symbol, allowing for the output of one process to communicate to another process through the pipe file, where the output of the one process is used as input for the other process.

```
root@shantanu-Dell-System-XPS-L502X:/dev# ls -l | grep ^p
prw-r--r-- 1 root root 0 Jul 19 00:28 GFG
prw-r--r-- 1 root root 0 Jul 19 00:28 GFG1
```

s: A socket file allows two processes to communicate, where both processes are allowed to either send or receive data.

```
root@shantanu-Dell-System-XPS-L502X:/dev# ls -l | grep ^s
srwxr-xr-x 1 root root 0 Jul 19 00:51 socket.sock
root@shantanu-Dell-System-XPS-L502X:/dev#
```

4.5 **Для чого використовуються дозволи Setuid та Setgid?

Setuid (Set User ID) and Setgid (Set Group ID) permissions are used to set special access rights for executable files in Unix-like systems.

- *Setuid*: If an executable file has the Setuid bit set and is owned by a user, it runs with the permissions of the file owner, rather than those of the user who executes it. This is useful for programs that require special privileges, such as programs that work with system resources like network configuration or password changes.
- *Setgid*: If an executable file has the Setgid bit set and belongs to a group, it runs with the permissions of the file group, rather than those of the user who executes it. This can be useful for files that require shared access to certain resources, for example, shared directories where users in a group can modify files owned by that group.

4.6 **Для чого в системі потрібен так званий “липкий біт” (Sticky Bit). Наведіть приклади коли цей дозвіл доцільно використовувати.

The Sticky Bit is used in the system to set special access rights for directories. The primary purpose of the Sticky Bit is to restrict the ability to delete or rename files by other users in a directory that has the Sticky Bit set.

A few examples of situations where using a sticky bit might be appropriate:

- The `/tmp`` directory: On many systems, the `/tmp`` directory is used as a temporary repository for files created by different users. Setting the sticky bit to `/tmp`` allows you to ensure that each user can only modify or delete their own files, not files created by other users.
- Public directories: If you have a directory in which files are frequently uploaded or created by many users, setting a sticky bit can prevent other users from accidentally deleting or moving files. This can be useful, for example, in web servers where files available for download are stored in a shared directory.
- Data protection: If you have a directory with sensitive data that should only be accessible by certain users, setting a sticky bit can ensure that files in that directory cannot be deleted or moved by other users.

Хід роботи:

1. Початкова робота в CLI-режимі в Linux ОС сімейства Linux:

- 1.1. Запустіть віртуальну машину VirtualBox, оберіть CentOS та запустіть її. Виконайте вхід в систему під користувачем: CentOS, пароль для входу: reverse (*якщо виконуєте ЛР у 401 ауд.*) та запустіть термінал.
- 1.2. Запустіть віртуальну машину Ubuntu_PC (*якщо виконуєте завдання ЛР через академію netacad*)
- 1.3. Запустіть свою операційну систему сімейства Linux (*якщо працюєте на власному ПК та її встановили*) та запустіть термінал.
2. Опрацюйте всі приклади команд, що представлені у лабораторних роботах курсу *NDG Linux Essentials: Lab 17: Ownership and Permissions* та *Lab 18: Special Directories and Files*. Створіть таблицю команд вивчених у п.2 ходу роботи у наступному вигляді.

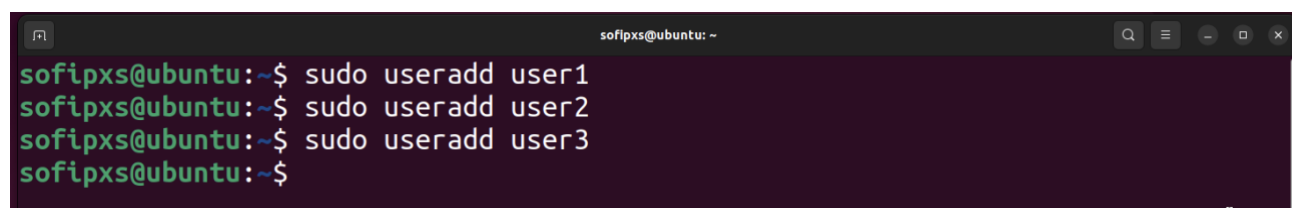
| Назва команди | Її призначення та функціональність |
|--|---|
| cd /tmp | Move to the /tmp directory |
| mkdir priv-dir pub-dir | Create two directories called priv-dir and pub-dir |
| touch priv-dir/priv-file touch pub-dir/pub-file | Create two files, one file called priv-file in the priv-dir directory and another file called pub-file in the pub-dir directory |
| ls -l priv-dir ls -l pub-dir | View the contents of the new directories. |
| ls -ld priv-dir/ | View permissions for the priv-dir directory |
| chmod o-rx priv-dir/ | Remove the others' permissions for read and execute |
| chmod a+x file | Gives everyone execute permission |
| chmod g-w file | Removes write permission for group owners |
| chmod go+r file | Adds read permission for group owner and others |
| chmod o=rwx | Sets others permissions to read, write and execute |
| chmod o+w pub-dir/ | Adds write permission for others to the pub-dir directory. |
| chmod g-rw,o-r priv-dir/priv-file | Removes read and write permissions for the group and read permission for others from the priv-file inside the priv-dir directory. |
| chmod a=rw pub-dir/pub-file | Sets the permissions for the owner, group, and others to read and write for the pub-file inside the pub-dir directory. |

| | |
|--|--|
| <code>echo "date" > test.sh</code> | Create a test.sh file in the /tmp containing the content "date" |
| <code>./test.sh</code> | Attempt to execute the test.sh file; it should fail. |
| <code>chmod u+x test.sh</code> | Adds execute permission for the owner (user) to the test.sh file. |
| <code>stat test.sh</code> | Verify the octal value for the permissions (access) to test.sh |
| <code>chmod 775 test.sh</code> | Sets the permissions for the owner, group, and others to read, write, and execute for the test.sh file. |
| <code>chown root:root pub-dir</code> | Change the user and group owner of pub-dir to the root user and the root group |
| <code>chown bin pub-dir/pub-file</code> | Change the user owner of the pub-file to the bin user |
| <code>ls -ld priv-dir</code> | View the details of the priv-dir and its contents |
| <code>ls -ld priv-dir</code> <code>ls -l priv-dir/priv-filechgrp -R users priv-dir</code> | To change the group ownership of all of the files of a directory structure, use the recursive -R option to the chgrp command. Change the group owner of the priv-dir and priv-file to the users group recursively with the chgrp command and view the updated files. |
| <code>ls -ld /tmp</code> | List the details of the /tmp directorie.Using the -d option for the ls command lists directory information; combined with the -l option it shows ownership and permissions for the directory files. |
| <code>ls -ld /var/tmp</code> | List the details of the /var/tmp directorie. |
| <code>ls -l /etc/shadow</code> | View the permissions on the /etc/shadow file. Specifically, the /etc/shadow file contains the encrypted passwords of all local user accounts and information about password aging (how long a password is valid). |
| <code>ls -l /usr/bin/passwd</code> | View the permissions of the /usr/bin/passwd file. Thus, the passwd command is able to update the /etc/shadow file, as it executes as the root user (recall that the root user can edit any file, regardless of the permissions on the file). |

| | |
|---|--|
| <code>ls -l /usr/bin/wall</code> | View the permissions on the /usr/bin/wall command |
| <code>cd</code> | Change to your home directory |
| <code>echo "data" > source</code> | Create a file named source containing the text "data" by using redirection |
| <code>ls -li source</code> | Using the -i option with the ls command prints the index number of the file. View the details and inode information of the source file |
| <code>ln source hardlink</code> | Create a hard link. |
| <code>ls -li source hardlink</code> | View the details and inode information of the source and new hard link file. |
| <code>ln hardlink hardlinktwo</code> <code>ls -li hardlink hardlinktwo</code> <code>source</code> | Create another hard link to the source file. View the details and inode information of the source and new hard link files. |
| <code>rm hardlinktwo</code> | Remove the last link that was created. |
| <code>ln -s source softlink</code> <code>ls -li source softlink</code> | The -s option for the ln command creates a symbolic link instead of a hard link. Create a symbolic link to the source file and view the details of both files. |
| <code>ln -s /proc crossdir</code> <code>ls -l crossdir</code> | Create a symbolic link to the /proc directory and display the link. |

3. Виконайте наступні практичні завдання у терміналі наступні дії (продемонструвати скріншоти):

- створіть трьох нових користувачів;



```

sofipxs@ubuntu: ~
sofipxs@ubuntu:~$ sudo useradd user1
sofipxs@ubuntu:~$ sudo useradd user2
sofipxs@ubuntu:~$ sudo useradd user3
sofipxs@ubuntu:~$

```

- створіть нову групу користувачів, туди додайте двох, з трьох створених користувачів;

```
sofipxs@ubuntu:~$ sudo groupadd newgroup
sofipxs@ubuntu:~$ sudo usermod -a -G newgroup user1 && sudo usermod -a -G newgroup user2
sofipxs@ubuntu:~$ getent group newgroup
newgroup:x:1010:user1,user2
sofipxs@ubuntu:~$
```

- створіть новий файл, який буде доступний на зчитування, редагування та виконання власником файлу, наприклад найпростіший скриптовий сценарій;

```
sofipxs@ubuntu:~$ nano script.sh
sofipxs@ubuntu:~$ chmod 700 script.sh
sofipxs@ubuntu:~$ ./script.sh
Hello, World!
sofipxs@ubuntu:~$ ls -l script.sh
-rwx----- 1 sofipxs sofipxs 34 Apr 22 19:56 script.sh
sofipxs@ubuntu:~$
```

- для користувачів групи власника надайте дозволи на перегляд та виконання (без дозволу на редагування) цього файлу;
- для інших користувачів заборонити доступ до цього файлу;

```
sofipxs@ubuntu:~$ chmod 750 script.sh
sofipxs@ubuntu:~$ ls -l script.sh
-rwxr-x--- 1 sofipxs sofipxs 34 Apr 22 19:56 script.sh
sofipxs@ubuntu:~$
```

- *подібні дії виконайте для директорій - створіть директорію, яка буде доступна для всіх трьох користувачів, створіть директорію, яку буде доступна тільки для власника, створіть директорію, яку користувачі групи власника зможуть переглядати, але не редагувати;

```
sofipxs@ubuntu:~$ mkdir private_directory
sofipxs@ubuntu:~$ chmod 700 private_directory
sofipxs@ubuntu:~$ ls -ld private_directory
drwx----- 2 sofipxs sofipxs 4096 Apr 22 20:40 private_directory
sofipxs@ubuntu:~$
```

```
sofipxs@ubuntu:~$ mkdir group_directory
sofipxs@ubuntu:~$ chmod 750 group_directory
sofipxs@ubuntu:~$ ls -ld group_directory
drwxr-x--- 2 sofipxs sofipxs 4096 Apr 22 20:44 group_directory
sofipxs@ubuntu:~$
```

```
sofipxs@ubuntu:~$ mkdir public_directory
sofipxs@ubuntu:~$ chmod 777 public_directory
sofipxs@ubuntu:~$ ls -ld public_directory
drwxrwxrwx 2 sofipxs sofipxs 4096 Apr 22 20:45 public_directory
sofipxs@ubuntu:~$
```

- *створить порожній файл під назвою emptyfile за допомогою команди touch emptyfile. Тепер “обнулить” дозволи для файлу з chmod 000 emptyfile. Що станеться, якщо змінити дозволи для emptyfile, передавши лише одне значення для chmod у числовому режимі, наприклад, chmod 4 emptyfile? Що буде, якщо ми використаємо два числа, наприклад chmod 44 emptyfile? Що ми можемо дізнатися про те, як chmod зчитує числове значення?

```
sofipxs@ubuntu:~$ touch emptyfile
sofipxs@ubuntu:~$ chmod 000 emptyfile
sofipxs@ubuntu:~$ ls -l emptyfile
----- 1 sofipxs sofipxs 0 Apr 22 20:47 emptyfile
sofipxs@ubuntu:~$
```

When we execute a chmod command with a single value in numeric mode, for example, chmod 4 emptyfile, it means that we are setting the read permission (r) for the owner of the file. In this case, only the owner of the file will have permission to read its contents. If we use two numbers, for example, chmod 44 emptyfile, it means that we are setting read permissions (r) for the file owner and the owner's group. In this case, both the owner of the file and the users in the corresponding group will have read permissions on the file.

```
sofipxs@ubuntu:~$ chmod 4 emptyfile
sofipxs@ubuntu:~$ ls -l emptyfile
-----r-- 1 sofipxs sofipxs 0 Apr 22 20:47 emptyfile
sofipxs@ubuntu:~$ chmod 44 emptyfile
sofipxs@ubuntu:~$ ls -l emptyfile
----r--r-- 1 sofipxs sofipxs 0 Apr 22 20:47 emptyfile
```

When we use the numeric method of the chmod command, we should be aware that each digit in the numeric value represents permissions for the owner, group, and other users in the order: owner, group, other. Each permitted action has a corresponding numeric value (e.g. 4 for read permission, 2 for write permission, 1 for execute permission). By adding these numbers, we can set the desired combinations of permissions for different users.

Робота студентки групи РПЗ-136 Дімітрової Софії

- **створіть каталог під назвою, де всі файли автоматично будуть належати Вашій групі користувачів і можуть бути видалені лише користувачем, який їх створив?

```
sofipxs@ubuntu:~$ mkdir mgrdirectory && sudo chown :newgroup mgrdirectory
sofipxs@ubuntu:~$ sudo chmod 2775 mgrdirectory
sofipxs@ubuntu:~$
```

```
sofipxs@ubuntu:~$ ls -ld mgrdirectory
drwxrwsr-x 2 sofipxs newgroup 4096 Apr 23 17:33 mgrdirectory
sofipxs@ubuntu:~$
```

2 in 2775 sets the setgid bit for a directory, which means that all new files and subdirectories created in that directory will belong to the group of that directory. The 775 indicates that the owner and group have full rights (read, write, and execute (rwx)) and that other users can read and execute(r-x). The s character indicates that the setuid or setgid bits are set. In our case, s is at runtime for the group (rws), which means that the setgid bit is set.

Instead of 2775, you can enter 2770, and then other users would not have access to the files at all.

- **під кожним користувачем створіть по одному новому файлу, та жорстке та символічне посилання на нього;

```
sofipxs@ubuntu:~$ cd mgrdirectory
sofipxs@ubuntu:~/mgrdirectory$ sudo -u user1 touch fileus1
sofipxs@ubuntu:~/mgrdirectory$ sudo -u user1 ln fileus1 fileus1_hard
sofipxs@ubuntu:~/mgrdirectory$ sudo -u user1 ln -s fileus1 fileus1_symlink
sofipxs@ubuntu:~/mgrdirectory$ ls
fileus1 fileus1_hard fileus1_symlink
sofipxs@ubuntu:~/mgrdirectory$
```

```
sofipxs@ubuntu:~/mgrdirectory$ sudo -u user2 touch fileus2
sofipxs@ubuntu:~/mgrdirectory$ sudo -u user2 ln fileus2 fileus2_hard
sofipxs@ubuntu:~/mgrdirectory$ sudo -u user2 ln -s fileus2 fileus2_symlink
sofipxs@ubuntu:~/mgrdirectory$ ls
fileus1      fileus1_symlink  fileus2_hard
fileus1_hard fileus2          fileus2_symlink
sofipxs@ubuntu:~/mgrdirectory$
```



```
sofipxs@ubuntu:~/mgrdirectory$ touch fileus3
sofipxs@ubuntu:~/mgrdirectory$ ln fileus3 fileus3_hard
sofipxs@ubuntu:~/mgrdirectory$ ln -s fileus3 fileus3_symlink
sofipxs@ubuntu:~/mgrdirectory$ ls
fileus1      fileus2      fileus3
fileus1_hard fileus2_hard fileus3_hard
fileus1_symlink fileus2_symlink fileus3_symlink
sofipxs@ubuntu:~/mgrdirectory$
```

```
sofipxs@ubuntu:~/mgrdirectory$ ls -l
total 0
-rw-rw-r-- 2 user1  newgroup 0 Apr 23 18:00 fileus1
-rw-rw-r-- 2 user1  newgroup 0 Apr 23 18:00 fileus1_hard
lrwxrwxrwx 1 user1  newgroup 7 Apr 23 18:02 fileus1_symlink -> fileus1
-rw-rw-r-- 2 user2  newgroup 0 Apr 23 18:04 fileus2
-rw-rw-r-- 2 user2  newgroup 0 Apr 23 18:04 fileus2_hard
lrwxrwxrwx 1 user2  newgroup 7 Apr 23 18:05 fileus2_symlink -> fileus2
-rw-rw-r-- 2 sofipxs newgroup 0 Apr 23 18:07 fileus3
-rw-rw-r-- 2 sofipxs newgroup 0 Apr 23 18:07 fileus3_hard
lrwxrwxrwx 1 sofipxs newgroup 7 Apr 23 18:10 fileus3_symlink -> fileus3
sofipxs@ubuntu:~/mgrdirectory$
```

- **спробуйте іншими користувачами переглянути ці файли;

```
sofipxs@ubuntu:~/mgrdirectory$ sudo -u user1 cat fileus2_symlink && sudo -u user2
cat fileus1
```

```
sofipxs@ubuntu:~/mgrdirectory$ sudo -u sofi cat fileus2
sofipxs@ubuntu:~/mgrdirectory$
```

- **спробуйте іншими користувачами видалити ці файли, зробіть висновки.

```
sofipxs@ubuntu:~/mgrdirectory$ sudo -u user1 rm -r fileus2
sofipxs@ubuntu:~/mgrdirectory$ sudo -u sofi rm fileus3
rm: remove write-protected regular empty file 'fileus3'?
sofipxs@ubuntu:~/mgrdirectory$ ls
fileus1      fileus1_symlink fileus2_symlink fileus3_hard
fileus1_hard fileus2_hard    fileus3         fileus3_symlink
```

As we can see, the file created by a user of the group was deleted, but the file created by a user who is not a member of the group was not deleted. Since the task was to prevent users who are not the owners of the files from deleting files, we can set a special bit on the directory. It's called a "sticky bit". Sticky bit will not affect existing files. However, any new files that will be created in the mgrdirectory directory will be

Робота студентки групи РПЗ-136 Дімітрової Софії

prevented from being deleted by other users unless they are the owner of the files or the directory.

```
sofipxs@ubuntu:~/mgrdirectory$ cd
sofipxs@ubuntu:~$ sudo chmod +t mgrdirectory
```

Sticky bit permission is displayed as a t in the execute part of other's permissions. Lowercase t means both sticky bit and execute is set.

```
sofipxs@ubuntu:~$ ls -ld mgrdirectory
drwxrwsr-t 2 sofipxs newgroup 4096 Apr 23 19:03 mgrdirectory
sofipxs@ubuntu:~$
```

```
sofipxs@ubuntu:~/mgrdirectory$ touch file2
sofipxs@ubuntu:~/mgrdirectory$ ls -l file2
-rw-rw-r-- 1 sofipxs newgroup 0 Apr 23 19:03 file2
sofipxs@ubuntu:~/mgrdirectory$ sudo -u sofi rm file2
rm: remove write-protected regular empty file 'file2'?
sofipxs@ubuntu:~/mgrdirectory$ ls
file2      fileus1_hard      fileus2_hard      fileus3      fileus3_symlink
fileus1    fileus1_symlink    fileus2_symlink    fileus3_hard
```

```
sofipxs@ubuntu:~/mgrdirectory$ sudo -u user1 rm file2
rm: cannot remove 'file2': Operation not permitted
sofipxs@ubuntu:~/mgrdirectory$
```

Відповіді на контрольні запитання:

1. Наведіть приклади зміни прав доступу символічним методом (Symbolic Method)?

- Change access rights for the file owner: *chmod u+rw filename*. This command grants the owner (u) read (r), write (w), and execute (x) permissions for lab10.txt.

```
sofipxs@ubuntu:~$ ls -l lab10.txt
-rw-rw-r-- 1 sofipxs sofipxs 0 Apr 22 16:36 lab10.txt
sofipxs@ubuntu:~$ chmod u+rw lab10.txt
sofipxs@ubuntu:~$ ls -l lab10.txt
-rwxrw-r-- 1 sofipxs sofipxs 0 Apr 22 16:36 lab10.txt
sofipxs@ubuntu:~$
```

- Change access rights for a group: *chmod g-w file.txt*. This command removes (-) write permission (w) for group (g) for lab10.txt.

```
sofipxs@ubuntu:~$ ls -l lab10.txt
-rwxrw-r-- 1 sofipxs sofipxs 0 Apr 22 16:36 lab10.txt
sofipxs@ubuntu:~$ chmod g-w lab10.txt
sofipxs@ubuntu:~$ ls -l lab10.txt
-rwxr--r-- 1 sofipxs sofipxs 0 Apr 22 16:36 lab10.txt
sofipxs@ubuntu:~$
```

- Combined change of access rights: *chmod u=rw,g+rx,o-w file.txt*. This command sets (=) read and write (rw) permissions for the owner (u), adds (+) read and execute (rx) permissions for the group (g), and removes (-) write permissions (w) for other users (o) for the lab10.txt.

```
sofipxs@ubuntu:~$ ls -l lab10.txt
-rwxr--r-- 1 sofipxs sofipxs 0 Apr 22 16:36 lab10.txt
sofipxs@ubuntu:~$ chmod u=rw,g+rx,o-w lab10.txt
sofipxs@ubuntu:~$ ls -l lab10.txt
-rw-r-xr-- 1 sofipxs sofipxs 0 Apr 22 16:36 lab10.txt
```

- Gives everyone execute permission: *chmod a+x file.txt*

```
sofipxs@ubuntu:~$ chmod a+x lab10.txt
sofipxs@ubuntu:~$ ls -l lab10.txt
-rwxr-xr-x 1 sofipxs sofipxs 0 Apr 22 16:36 lab10.txt
sofipxs@ubuntu:~$
```

2. Наведіть приклади зміни прав доступу числовим методом (numeric method, octal method)?

In the numeric method, each of the access rights (read, write, execute) is represented by a three-digit number, where each digit corresponds to the access rights for the owner, group, and other users.

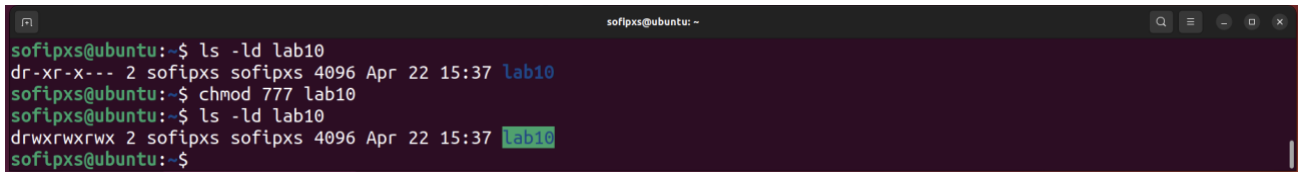
- Set access rights to "read and write" for the owner, "read" for the group and "no access" for the others: *chmod 640 file.txt*

```
sofipxs@ubuntu:~$ ls -ld lab10
drwxrwxr-x 2 sofipxs sofipxs 4096 Apr 22 15:37 lab10
sofipxs@ubuntu:~$ chmod 640 lab10
sofipxs@ubuntu:~$ ls -ld lab10
drw-r----- 2 sofipxs sofipxs 4096 Apr 22 15:37 lab10
sofipxs@ubuntu:~$
```

- Set access rights to "read and execute" for owner and group, "read" for others: *chmod 550 file.txt*

```
sofipxs@ubuntu:~$ ls -ld lab10
drwxrwxr-x 2 sofipxs sofipxs 4096 Apr 22 15:37 lab10
sofipxs@ubuntu:~$ chmod 550 lab10
sofipxs@ubuntu:~$ ls -ld lab10
dr-xr-xr-x 2 sofipxs sofipxs 4096 Apr 22 15:37 lab10
sofipxs@ubuntu:~$
```


- Set "read, write and execute" access rights for all: *chmod 777 file.txt*

A terminal window with a dark purple background. The prompt is 'sofipxs@ubuntu: ~'. The user enters 'ls -ld lab10', and the output is 'dr-xr-x--- 2 sofipxs sofipxs 4096 Apr 22 15:37 lab10'. Then the user enters 'chmod 777 lab10'. Finally, the user enters 'ls -ld lab10' again, and the output is 'drwxrwxrwx 2 sofipxs sofipxs 4096 Apr 22 15:37 lab10'.

```
sofipxs@ubuntu:~$ ls -ld lab10
dr-xr-x--- 2 sofipxs sofipxs 4096 Apr 22 15:37 lab10
sofipxs@ubuntu:~$ chmod 777 lab10
sofipxs@ubuntu:~$ ls -ld lab10
drwxrwxrwx 2 sofipxs sofipxs 4096 Apr 22 15:37 lab10
sofipxs@ubuntu:~$
```

3. Яке призначення команди umask?

The purpose of the *umask* command is to set the default file creation permissions mask. It determines which permissions are automatically subtracted from the maximum permissions when a new file or directory is created. It helps control the default permissions to enhance security and privacy.

4. Порівняйте жорсткі та символічні посилання?

- A *hard link* is a direct link to a file by its inode index number. A hard link is created to an existing file on the same file system and looks like an additional name for that file. All hard links point to the same file, and they cannot point to directories or files on other file systems.
- A *symbolic link* is a reference to a file or directory that contains the path to the target file. A symbolic link allows you to create links to files or directories that may be located elsewhere in the file system or even on other file systems.

5. *Чи можна виконати файл, для якого є права на виконання, але не встановлені права на читання (--x)? Поясніть.

No. It is not possible to execute a file that has execute privileges but not read privileges (--x). To execute a file, you must have at least read permission, because the operating system must read the contents of the file to execute its contents. If you do not have read permission (--x), the operating system cannot read the file and execute it.

6. *Якщо ми змінюємо права доступу та дозволи в поточній сесії чи будуть вони збережені в наступній?

No. Changes to access rights and permissions made in the current session are not saved for subsequent sessions by default. When the session is closed or the system is restarted, the changed access rights and permissions return to the values set in the file system configuration or default settings. To make changes to access rights permanent and save them between sessions, you need to make appropriate changes to the configuration files or use other mechanisms, such as scripts or automated procedures, to apply the changes at each startup.

7. *Чи є якийсь шаблон, яким система користується щодо прав та доступів при створенні нових файлів. Як можна змінити права дозволу за замовчуванням?

Yes, Linux has a template that the system uses to set default permissions when creating new files. This pattern is called the “umask” (User File Creation Mask). The umask determines which permission bits are disabled by default when a new file is created. Relatively speaking, umask determines which access rights will not be granted to the new file. The umask value is calculated from the established rules and can be displayed as an octal number.

To change the default permissions, you can use the ``umask`` command on the command line or set the umask value in the appropriate configuration files (for example, `~/.bashrc` or `/etc/profile`). If you want to set the permissions to 644 (rw-r--r--) as the umask value, you can run the ``umask 022`` command or add ``umask 022`` to the appropriate configuration file.

Example:

- To make changes to the mask permanent, open the `~/.bashrc` file in a text editor:
`nano ~/.bashrc`.
- Add the line `umask 022` to the end of the `~/.bashrc` file. Save your changes and close the `~/.bashrc` file.
- To apply the new umask value without rebooting, run the command:
`source ~/.bashrc`.

The default permissions for new files created on your system will now be set to 644 (rw-r--r--).

8. *Яким чином можна створити жорстке посилання? В яких ситуаціях їх доцільно використовувати?

To create hard links, the `ln` command is used with two arguments. The first argument is an existing file name to link to, called a target, and the second argument is the new file name to link to the target: `ln target link_name`

Hard links are useful when you need to have multiple names pointing to the same file. They maintain links to the same file system node structure, so changes to one name are reflected in all other links. Hard links cannot be created for directories and files that are on different file systems.

9. *Яким чином можна створити символічне посилання? В яких ситуаціях їх доцільно використовувати?

To create a symbolic link, you can use the `ln` command with the `-s` option followed by the target file or directory and the desired name for the symbolic link:
`ln -s target link_name`

Symbolic links are appropriate to use when you want a file or directory to be accessible from a different location or when you want to create shortcuts or references to files. Symbolic links can point to files or directories across different file systems.

10.**Уявіть, що програмі потрібно створити одноразовий тимчасовий файл, який більше ніколи не знадобиться після закриття програми. Який правильний каталог для створення цього файлу?

To create a one-time temporary file that will no longer be needed after the program is closed, it is recommended to use a special directory for temporary data. On many Unix systems, including Linux, this directory is called `/tmp`.

The `/tmp` directory is intended specifically for temporary files that are used during program execution and are no longer needed after the program is closed. Various temporary files are stored in this directory, such as caches, intermediate results of calculations, temporary files created by programs, etc.

11.**Є файл оригінал та для нього створено два посилання - символічне та жорстке. Що відбудеться з іншими файлами, якщо видалити:

- файл оригінал;

The file will be permanently deleted. Symbolic links will become broken, and hard links will still be able to access the file's content until they are deleted or the system is rebooted.

- символічне посилання;

Only the symbolic link itself will be deleted. The original file and other hard links will remain unaffected.

- жорстке посилання.

The hard link will be deleted, but the original file and other hard links will still retain the file's content until they are deleted or the system is rebooted. The symbolic link will become invalid because it was referring to a hard link that has been removed. Using a symbolic link will cause a "Link broken" or "File does not exist" error.

Висновки:

In the course of the laboratory work, various aspects of access rights to files and directories in the Linux system were investigated. The following topics were discussed in detail: viewing file owner access rights, changing the owner, viewing file type, Setuid and Setgid permissions, and the concept of "sticky bit". Practical skills in working with the `chmod`, `chown`, `ln` commands were acquired.