“Kyiv Vocation College of Communication”
Cyclical Commission of Computer Engineering

REPORT ON EXECUTION LABORATORY WORK №10

on the discipline: "Operating Systems"

Topic: "Changing File Owners and Permissions in Linux. Special Directories and Files in Linux"

Performed by student
group RPZ-13a
Kateryna Hranat
Checked by teacher
V.S. Sushanova

Performed by student group RPZ-13a Kateryna Hranat

Kyiv 2024

Work objectives:

1. Gaining Practical Skills in Working with the Bash Command Shell.
2. Familiarization with Basic Actions for Changing File Owners and File Permissions.
3. Understanding Special Directories and Files in Linux.

Material Support for Classes:

1. IBM PC type computers.
2. Windows operating system and VirtualBox virtual machine (Oracle).
3. GNU/Linux operating system (any distribution).
4. Cisco Networking Academy website netacad.com and its online courses on Linux.

Performed by student group RPZ-13a Kateryna Hranat

Tasks for Preliminary Preparation:

1. Read the brief theoretical information for the laboratory work and create a small dictionary of basic English terms related to the purpose of commands and their parameters.

English terms	**Ukrainian terms**
File Ownership	Власність на файли
Changing Groups	Зміна груп
Supplemental groups	Додаткові групи
Circumstances	Обставини
Hidden files	Приховані файли

2. Study the materials of the online course "NDG Linux Essentials" from Cisco:
   - Chapter 17 - Ownership and Permissions
   - Chapter 18 - Special Directories and Files

Complete✔

3. Complete testing in the NDG Linux Essentials course on the following topics:
   - Chapter 17 Exam
   - Chapter 18 Exam

Complete✔

4. Based on the material covered, provide answers to the following questions:

4.1 What is the purpose of the ‘id’ command?

The ‘id’ command is used to display user and group identity information, including the user ID (UID), group ID (GID), and any supplemental groups to which the user belongs.

4.2 How to view what permissions the file owner has?

To view the permissions of a file owner, you can use the ‘ls -l’ command, which lists files and their permissions. The output will show who the file

owner is and what permissions they have, indicated by a sequence of letters and dashes, e.g., ‘rwxr-xr-x’.
4.3 How to change the group owner?
To change the group owner, you can use the ‘chgrp’ command followed by the group name and the file name, e.g., ‘chgrp newgroup filename’.
4.4 How to check in the terminal what type of a file it is? Provide examples for different file types.
To identify the type of a ‘file’, you can use the file command, followed by the file name. It provides information about the file type, e.g., whether it's a text file, a directory, an executable, or a symbolic link.
Examples:

- For a text file, the command might return "ASCII text".
- For a directory, it might return "directory".
- For an executable file, it might return "executable".

4.5 What is the purpose of the Setuid and Setgid permissions?
Setuid (Set User ID) and Setgid (Set Group ID) are special permissions in Linux that allow a program to run with the permissions of its owner or group, rather than the permissions of the user running the program. This is useful for operations that require elevated privileges, such as accessing system resources.
4.6 What is the "sticky bit" and when is it appropriate to use it?
The "sticky bit" is a special permission that, when set on a directory, allows only the file's owner (or the root user) to delete or modify its contents, even if other users have write permissions in the directory. This is often used in shared directories like ‘/tmp’, where multiple users have write access, but you want to prevent them from deleting or modifying each other's files.

5. Prepare an initial version of the report in electronic form:
    - Title page, topic, and purpose of the work
    - Glossary of terms
    - Answers to points 4.1 - 4.6 from the tasks for preliminary preparation Complete✔

Performed by student group RPZ-13a Kateryna Hranat

Progress of Work:

1. Initial work in CLI mode in a Linux operating system of the Linux family:

   <u>1.1. Start your Linux-based operating system (if you are using your own PC and have it installed) and open the terminal.</u>

   Work through all the command examples provided in the lab assignments of the NDG Linux Essentials: Lab 17: Ownership and Permissions та Lab 18: Special Directories and Files. Create a table to describe these commands.

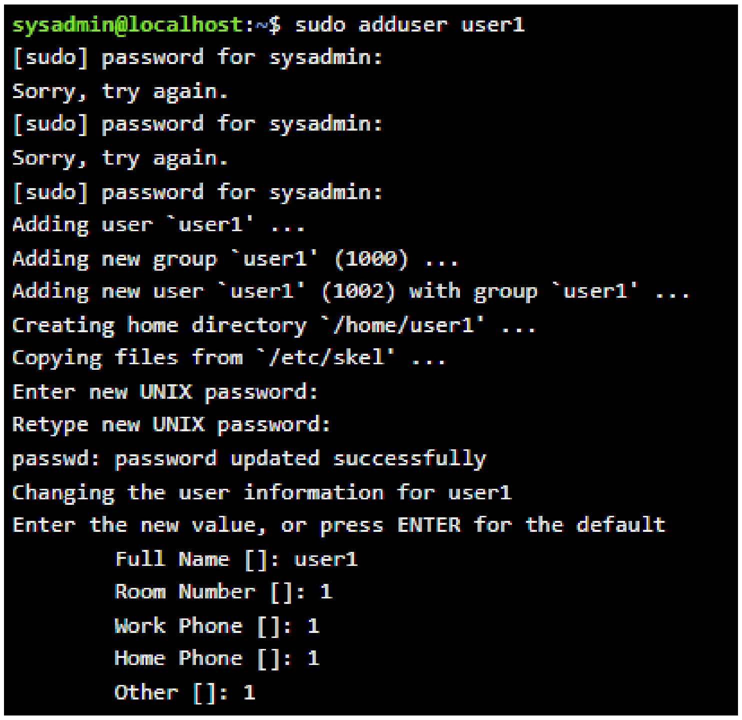
Command	**Description**
\`id\`	Displays user ID, group ID, and all supplementary group IDs for a specified user.
\`chown\`	Changes the owner of a file or directory. Accepts the new owner's user ID or username as an argument.
\`chgrp\`	Changes the group ownership of a file or directory. Requires the new group name or group ID.
\`chmod\`	Modifies the permissions of a file or directory. Uses numeric (e.g., \`755\`) or symbolic (e.g., \`u+x\`) representations for permissions.
\`ls -l\`	Lists files and directories with detailed information, including permissions, owner, group, and size.
\`file\`	Determines the file type and displays it, which can be helpful to distinguish between different file formats.

`umask`	Sets the default permissions mask for newly created files or directories. Controls which permissions are automatically removed when creating files.
`setuid`	A special permission setting that allows executable files to run with the file owner's permissions. Commonly used for programs that require elevated privileges.
`setgid`	A special permission setting that allows executable files to run with the group owner's permissions. It can also be set on directories to ensure new files inherit the group ownership.
`sticky bit`	A special permission that, when set on a directory, only allows the owner of a file to delete or rename it, regardless of other permissions. Typically used on shared directories like `/tmp`.

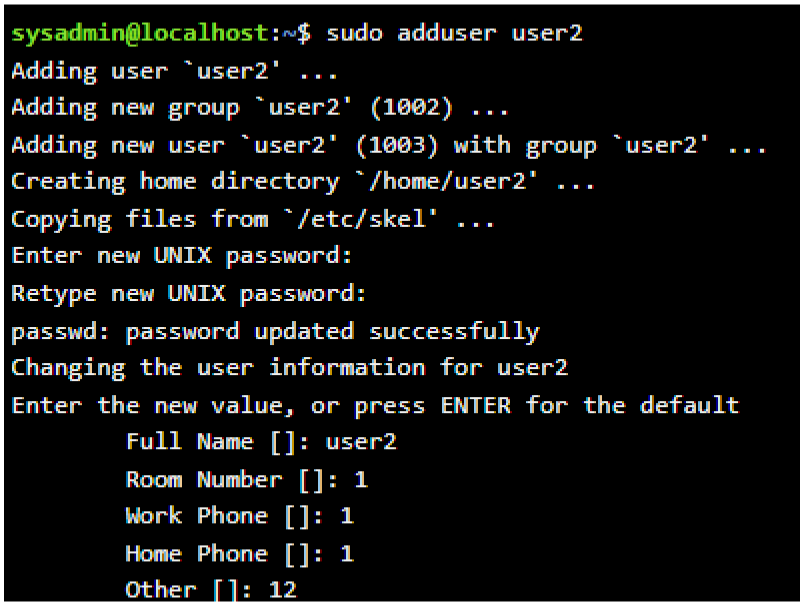
2. Execute practical tasks in the terminal (demonstrate screenshots):

Task: Create Three New Users

Performed by student group RPZ-13a Kateryna Hranat

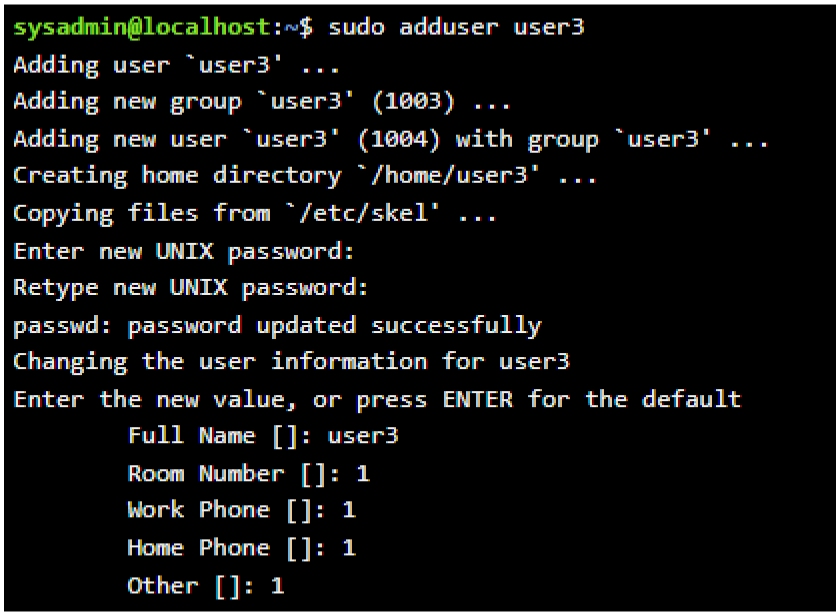


```
sysadmin@localhost:~$ sudo adduser user1
[sudo] password for sysadmin:
Sorry, try again.
[sudo] password for sysadmin:
Sorry, try again.
[sudo] password for sysadmin:
Adding user `user1' ...
Adding new group `user1' (1000) ...
Adding new user `user1' (1002) with group `user1' ...
Creating home directory `/home/user1' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for user1
Enter the new value, or press ENTER for the default
        Full Name []: user1
        Room Number []: 1
        Work Phone []: 1
        Home Phone []: 1
        Other []: 1
```



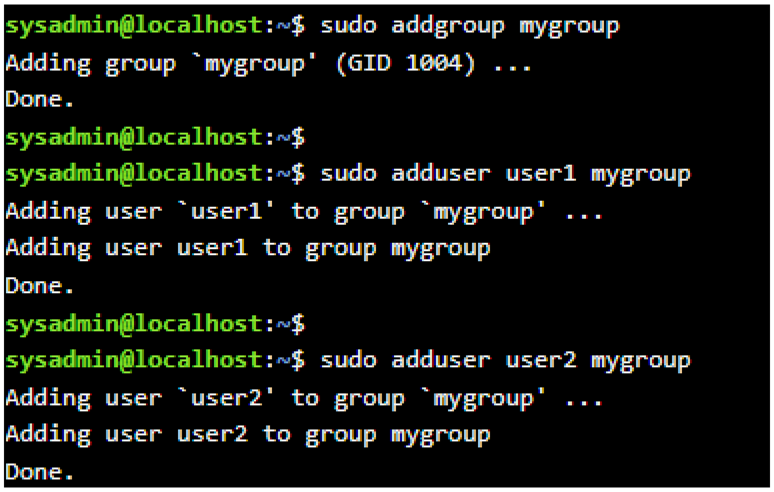
```
sysadmin@localhost:~$ sudo adduser user2
Adding user `user2' ...
Adding new group `user2' (1002) ...
Adding new user `user2' (1003) with group `user2' ...
Creating home directory `/home/user2' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for user2
Enter the new value, or press ENTER for the default
        Full Name []: user2
        Room Number []: 1
        Work Phone []: 1
        Home Phone []: 1
        Other []: 12
```

Performed by student group RPZ-13a Kateryna Hranat



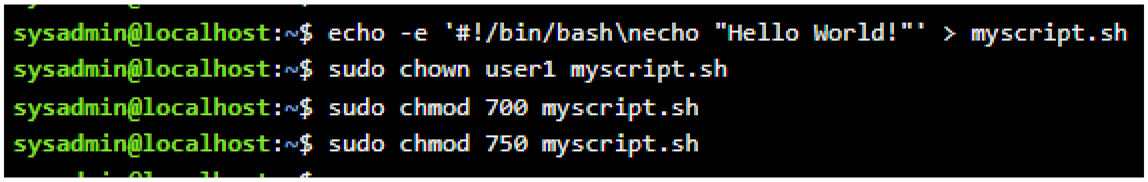
```
sysadmin@localhost:~$ sudo adduser user3
Adding user `user3' ...
Adding new group `user3' (1003) ...
Adding new user `user3' (1004) with group `user3' ...
Creating home directory `/home/user3' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for user3
Enter the new value, or press ENTER for the default
        Full Name []: user3
        Room Number []: 1
        Work Phone []: 1
        Home Phone []: 1
        Other []: 1
```

Task: Create a New User Group and Add Two Users



```
sysadmin@localhost:~$ sudo addgroup mygroup
Adding group `mygroup' (GID 1004) ...
Done.
sysadmin@localhost:~$
sysadmin@localhost:~$ sudo adduser user1 mygroup
Adding user `user1' to group `mygroup' ...
Adding user user1 to group mygroup
Done.
sysadmin@localhost:~$
sysadmin@localhost:~$ sudo adduser user2 mygroup
Adding user `user2' to group `mygroup' ...
Adding user user2 to group mygroup
Done.
```

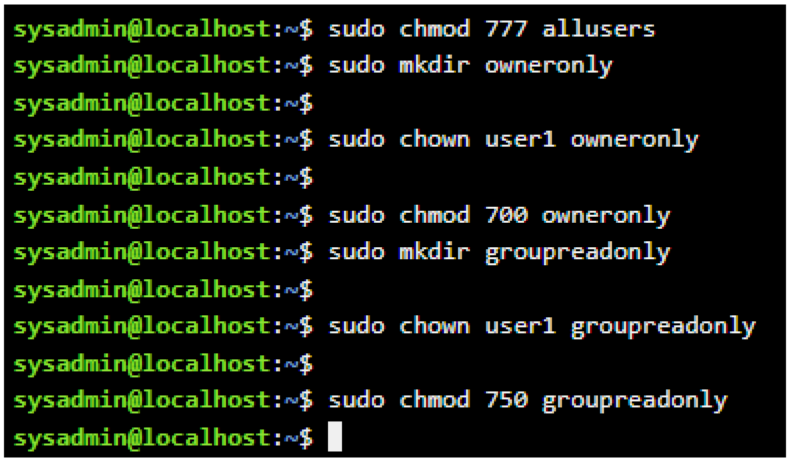
Task: Create a File and Modify Permissions



```
sysadmin@localhost:~$ echo -e '#!/bin/bash\necho "Hello World!"' > myscript.sh
sysadmin@localhost:~$ sudo chown user1 myscript.sh
sysadmin@localhost:~$ sudo chmod 700 myscript.sh
sysadmin@localhost:~$ sudo chmod 750 myscript.sh
```

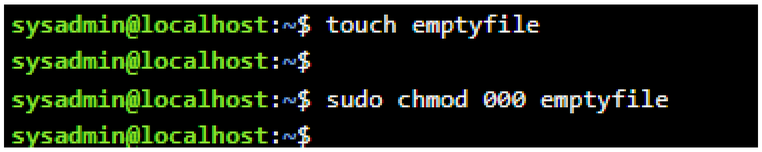
Performed by student group RPZ-13a Kateryna Hranat

Task: Create Directories with Different Permissions

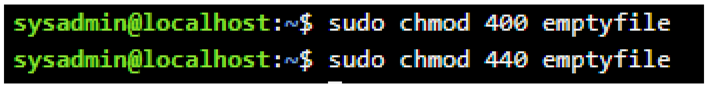


```
sysadmin@localhost:~$ sudo chmod 777 allusers
sysadmin@localhost:~$ sudo mkdir owneronly
sysadmin@localhost:~$
sysadmin@localhost:~$ sudo chown user1 owneronly
sysadmin@localhost:~$
sysadmin@localhost:~$ sudo chmod 700 owneronly
sysadmin@localhost:~$ sudo mkdir groupreadonly
sysadmin@localhost:~$
sysadmin@localhost:~$ sudo chown user1 groupreadonly
sysadmin@localhost:~$
sysadmin@localhost:~$ sudo chmod 750 groupreadonly
sysadmin@localhost:~$
```

Task: Modify Permissions of a File with `chmod 000` and Observe

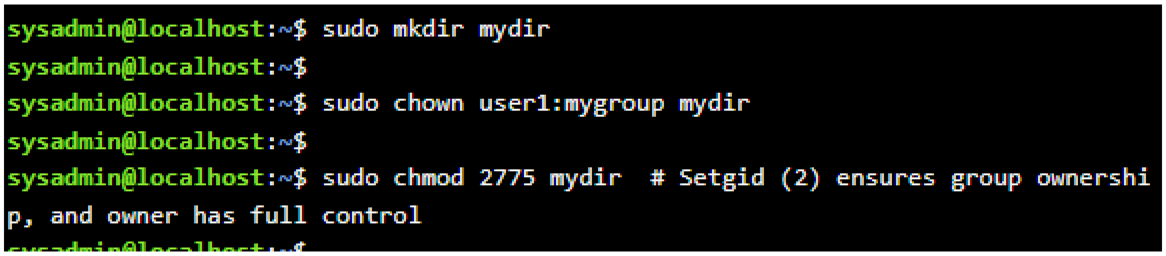


```
sysadmin@localhost:~$ touch emptyfile
sysadmin@localhost:~$
sysadmin@localhost:~$ sudo chmod 000 emptyfile
sysadmin@localhost:~$
```



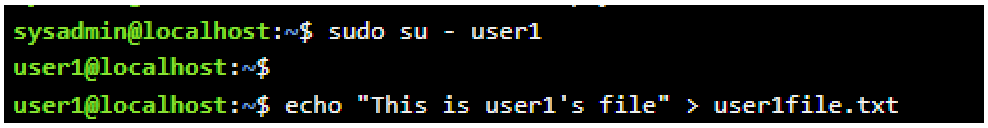
```
sysadmin@localhost:~$ sudo chmod 400 emptyfile
sysadmin@localhost:~$ sudo chmod 440 emptyfile
```

Task: Create Directory with Setgid

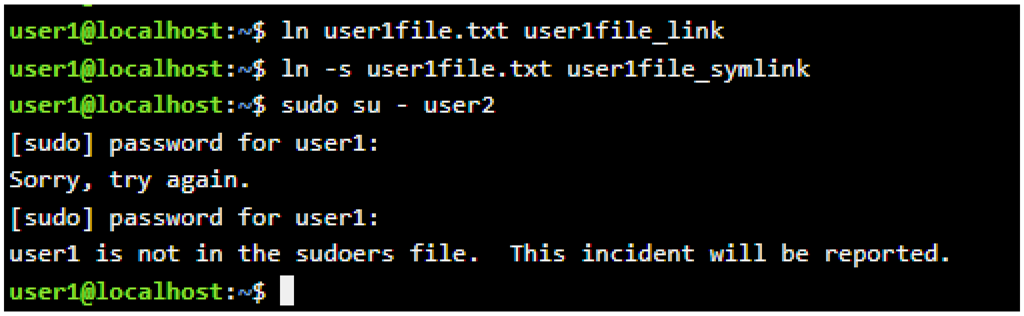


```
sysadmin@localhost:~$ sudo mkdir mydir
sysadmin@localhost:~$
sysadmin@localhost:~$ sudo chown user1:mygroup mydir
sysadmin@localhost:~$
sysadmin@localhost:~$ sudo chmod 2775 mydir  # Setgid (2) ensures group ownershi
p, and owner has full control
```

Task: Create Files and Hard and Symbolic Links



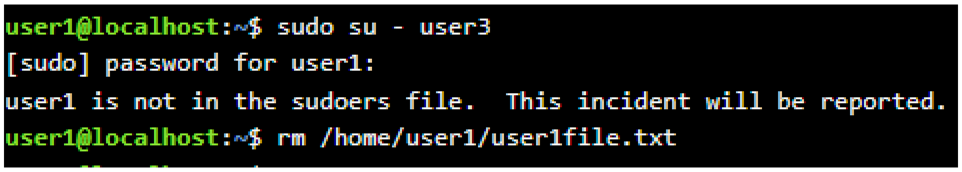
```
sysadmin@localhost:~$ sudo su - user1
user1@localhost:~$
user1@localhost:~$ echo "This is user1's file" > user1file.txt
```



```
user1@localhost:~$ ln user1file.txt user1file_link
user1@localhost:~$ ln -s user1file.txt user1file_symlink
user1@localhost:~$ sudo su - user2
[sudo] password for user1:
Sorry, try again.
[sudo] password for user1:
user1 is not in the sudoers file.  This incident will be reported.
user1@localhost:~$
```

Performed by student group RPZ-13a Kateryna Hranat

Task: Attempt to Delete Files Created by Other Users



```
user1@localhost:~$ sudo su - user3
[sudo] password for user1:
user1 is not in the sudoers file.  This incident will be reported.
user1@localhost:~$ rm /home/user1/user1file.txt
```

Control questions

1. Provide examples of changing file permissions using the symbolic method.
   - To add execution permission for the owner, use `chmod u+x filename`.
   - To remove write permission for the group, use `chmod g-w filename`.
   - To add read permission for others, use `chmod o+r filename`.
2. Provide examples of changing file permissions using the numeric (octal) method.
   - To set full permissions for the owner, read and execute for group, and no permissions for others, use `chmod 750 filename`.
   - To set read-only for all users, use `chmod 444 filename`.
   - To set full permissions for everyone, use `chmod 777 filename`.
3. What is the purpose of the `umask` command?
   - The `umask` command sets the default permissions for newly created files and directories. It acts as a mask that subtracts permissions from the maximum possible (usually 777 for directories and 666 for files). This way, `umask` controls what permissions are granted when new files or directories are created.
4. Compare hard links and symbolic links.
   - Hard Links: Share the same inode and storage space as the original file. If you delete the original file, the hard link still retains the data. Suitable for keeping redundant access to the same data.

- Symbolic Links: Point to the path of the target file. If the original file is deleted, the symbolic link becomes invalid. Useful for creating shortcuts to other files or directories.

5. *Can you execute a file that has execution rights but no read rights (--x)? Explain.

   - In general, you cannot execute a file that doesn't have read permissions. The operating system needs to read the file to execute it. Thus, execution requires both read and execute permissions.

6. *If we change file permissions and access rights in the current session, will they persist in the next session?

   - Yes, file permissions and access rights changes are persistent across sessions. Once you change the permissions, they remain until changed again, regardless of whether you log out or restart the system.

7. *Is there a system-wide pattern for file permissions when creating new files? How can default permissions be changed?

   - Default permissions for new files are governed by the system's `umask`. You can view or set it with the `umask` command. By default, the `umask` for files is often `022`, which allows read/write for the owner, read-only for others. To change the default permissions, update the `umask` value.

8. *How can you create a hard link? In what situations would you use hard links?

   - To create a hard link, use `ln sourcefile hardlink`. Hard links are suitable when you want multiple references to the same file without redundancy. If the original is removed, hard links keep the content alive.

9. *How can you create a symbolic link? In what situations would you use symbolic links?

   - To create a symbolic link, use `ln -s sourcefile symlink`. Symbolic links are helpful for creating shortcuts to files or directories and for pointing to resources that might move or change.

10. **If a program needs to create a temporary file that will no longer be needed after the program closes, which directory is the most suitable?

  - The appropriate directory for temporary files that are no longer needed after program closure is `/tmp`. This directory is designed for temporary storage and is usually cleaned upon reboot.

11. **Suppose there's an original file with two links—a symbolic link and a hard link. What happens if you delete: the original file; the symbolic link; the hard link?

   - Deleting the Original File: Hard links retain the file data, while symbolic links become invalid ("broken").
   - Deleting the Symbolic Link: The original file and any hard links remain unaffected.
   - Deleting the Hard Link: The data still exists in the original file or other hard links. If it's the last hard link, the data is lost.

Conclusions:

In conclusion, I have better gaining practical skills in working with the Bash Command Shell. Familiarization with basic actions for changing File Owners and File Permissions. Understanding special directories and Files in Linux. In addition, when permissions are set properly, users and groups have restricted access, preventing accidental or unauthorized modifications. Setgid can enforce group ownership, while the sticky bit protects against unwanted deletions. Hard links share the same inode, so deleting one does not affect the other, while symbolic links point to the target file and become invalid if it is removed.