Operating System

Lab – 11

Objectives:

1. Understanding the concept of Hard and Soft links, their differences and limitations.
2. Understanding the concept of permissions and controlling user access on files.

Resources:

- Video Lecture 21: https://www.youtube.com/watch?v=g8xZgtuYiWI&spfreload=10
- Video Lecture 22: https://www.youtube.com/watch?v=tEYYasCVxRc

Task 1:

- **What are hard and soft links? Elaborate your answer to TA by drawing a comprehensive diagram.**

Hard Links and Soft Links

- **Hard Links**:
  - A hard link is a direct reference (alias) to the inode of a file on a filesystem.
  - Both the original file and the hard link share the same inode number and point to the same data blocks.
  - Hard links only work within the same filesystem.
- **Soft Links (Symbolic Links)**:
  - A soft link is a pointer to the file name and path of the original file.
  - It is a separate file with a different inode number.
  - Soft links can span across different filesystems.

- **What are the limitations and differences between hard and soft links? Explain your answer with help of set of commands and explain it to TA.**

Feature/Aspect	Hard Link	Soft Link
Inode Number	Same as the original file	Different from the original file
Filesystem Restriction	Must be on the same filesystem	Can span across different filesystems
Original File Deletion	Does not affect accessibility	Becomes inaccessible
Directory Linking	Not allowed	Allowed (but rarely used)
File Size	Same as the file	Small (stores file path only)

- **What happen if we delete the original file? Is it still accessible from its hard link? Justify your answer.**

  Yes, the file remains **accessible** from its hard link even after the original file is deleted.

  Hard Link Characteristics:

  - A hard link points to the same inode as the original file.
  - In a filesystem, the inode stores the file's metadata (e.g., location of data blocks, file size, permissions).
  - When you delete the "original file," you are only removing one reference to the inode. The data still exists because at least one reference (the hard link) remains.

- **What happen if we delete the original file? Is it still accessible from its soft link? Justify your answer.**

  No, the file is **not accessible** from its soft link if the original file is deleted.

- **Soft Link Characteristics**:

  - When you delete the original file, the path stored in the soft link becomes invalid (a "broken link").
  - Since the soft link does not store the actual data, it cannot provide access to the file content.
  - A soft link (symbolic link) is a separate file that stores the path of the original file as a reference.

- **Why the inode numbers of original file and its hard link are same where soft link has different inode from original file?**

  Hard Links

  Reason: Hard links share the same inode as the original file because they are just additional directory entries pointing to the same inode.

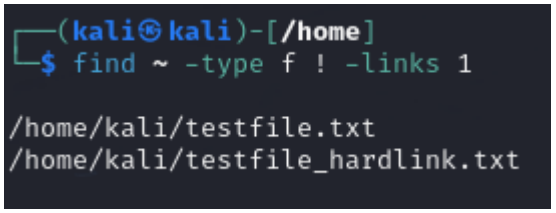
  - In the filesystem, the inode contains metadata (file size, permissions, data block pointers) and keeps track of how many directory entries point to it (link count).
  - Creating a hard link simply adds another directory entry pointing to the same inode.

  Soft Links

  Reason: A soft link (symbolic link) is a separate file with its own inode.

  - It stores the pathname of the target file as its content rather than pointing directly to the original file's inode.
  - Soft links are essentially shortcut files, and they do not directly reference the original file's metadata or data blocks.

- **Use `find` to look in your home directory for regular files that do not (!) have one hard link.**



Task 2:

- **As normal user, create a directory `~/permissions`. Create a file owned by yourself in this directory. Copy a file owned by root from `/etc/passwd` to your permissions dir, who owns this file now and why?**

```
┌──(kali㉿kali)-[/home]
└─$ mkdir ~/permissions

┌──(kali㉿kali)-[/home]
└─$ echo "This is my file" > ~/permissions/myfile.txt

┌──(kali㉿kali)-[/home]
└─$ cp /etc/passwd ~/permissions/

┌──(kali㉿kali)-[/home]
└─$ ls -l ~/permissions/

total 8
-rw-rw-r-- 1 kali kali   16 Dec  8 05:46 myfile.txt
-rw-r--r-- 1 kali kali 3379 Dec  8 05:47 passwd
```

Who Owns the File Now and Why?

- **Ownership of the Copied File**: The copied file is now owned by the current user (`kali`), not `root`.
- **Reason**: When you copy a file, the system creates a new file in the destination directory. By default, the new file is owned by the user performing the copy operation, and the file's metadata (ownership and group) is not preserved unless you explicitly use the `--preserve` option.

- **As root, create a file in the user's `~/permissions` directory. As normal user, look at who owns this file created by root.**

```
┌──(kali㉿kali)-[/home]
└─$ sudo -i

┌──(root㉿kali)-[~]
└─# echo "File created by root" > /home/kali/permissions/rootfile.txt

┌──(root㉿kali)-[~]
└─# ls -l /home/kali/permissions/rootfile.txt

-rw-r--r-- 1 root root 21 Dec  8 05:49 /home/kali/permissions/rootfile.txt

┌──(root㉿kali)-[~]
└─# su kali
zsh: corrupt history file /home/kali/.zsh_history
┌──(kali㉿kali)-[/root]
└─$ ls -l ~/permissions/rootfile.txt

-rw-r--r-- 1 root root 21 Dec  8 05:49 /home/kali/permissions/rootfile.txt
```

The file is owned by `root` because ownership is determined by the user who creates the file.
The user (`kali`) cannot modify or delete the file without elevated privileges unless granted ownership or proper permissions.

- **Change the ownership of all files in `~/permissions` to yourself. Make sure you have all rights to these files, and others can only read.**

```
┌──(kali㉿kali)-[/root]
└─$ sudo chown -R $(whoami):$(whoami) ~/permissions

[sudo] password for kali:

┌──(kali㉿kali)-[/root]
└─$ chmod -R 744 ~/permissions


┌──(kali㉿kali)-[/root]
└─$ ls -l ~/permissions

total 12
-rwxr--r-- 1 kali kali   16 Dec  8 05:46 myfile.txt
-rwxr--r-- 1 kali kali 3379 Dec  8 05:47 passwd
-rwxr--r-- 1 kali kali   21 Dec  8 05:49 rootfile.txt
```

- **With chmod, is 734 the same as `rwxr-xr--`?**
  NO, its **rwx-wxr--** (734)

  For 734:

  1. **7** (Owner): `rwx` → Read, write, and execute.
  2. **3** (Group): `-wx` → Write and execute, no read.
  3. **4** (Others): `r--` → Read-only.

  Result: 734 translates to `rwx-wx-r--`.

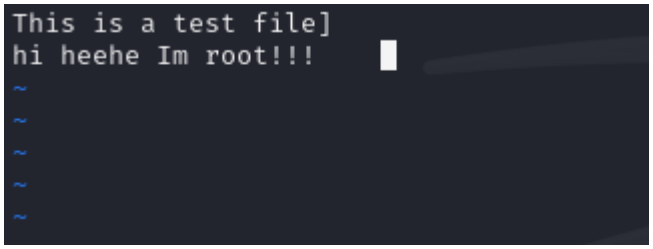
- **Create a file as normal user, give only read to others. Can another normal user read this file? Test writing to this file with vim. Can root read this file? Can root write to this file with vim?**

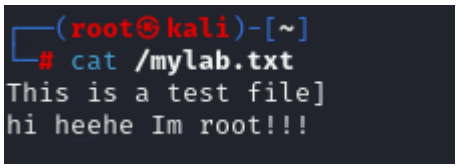
```
┌──(kali㉿kali)-[/]
└─$ sudo echo "This is a test file" > mylab.txt


┌──(kali㉿kali)-[/]
└─$ chmod 644 mylab.txt
```

```
$ sudo cat /home/kali/mylab.txt
[sudo] password for other:
Sorry, try again.
[sudo] password for other:
other is not in the sudoers file.
$ sudo cat /mylab.txt
[sudo] password for other:
other is not in the sudoers file.
$ vim /mylab.txt
```

```
-- INSERT -- W10: Warning: Changing a readonly file
```

```
┌──(kali㉿kali)-[/]
└─$ sudo echo "This is a test file" > mylab.txt


┌──(kali㉿kali)-[/]
└─$ chmod 644 mylab.txt
```





- **In order to copy f1.txt from `/d1/d2/d3/f1.txt` to `d4/d5/d6/` directory what should be the minimum permissions on these directories and why?**

1. Source Directory (`/d1/d2/d3/`)

To read the file `f1.txt` from `/d1/d2/d3/`, the following permissions are required on each directory in the path:

- **Permissions on `/d1`, `/d1/d2`, `/d1/d2/d3`**: Execute (`x`) to traverse.
- **Permissions on `/d1/d2/d3/`**: Read (`r`) to list files.
- **Permissions on `f1.txt`**: Read (`r`) to read the file content.

2. Destination Directory (`d4/d5/d6/`)

To copy the file into `d4/d5/d6/`, the following permissions are required:

- **Permissions on `/d4`, `/d4/d5`, `/d4/d5/d6`**: Execute (`x`) to traverse.
- **Permissions on `d4/d5/d6/`**: Write (`w`) and Execute (`x`) to create the new file.

Explanation of Why These Permissions Are Needed

- **Read (`r`)**: Needed to read file contents and list files in a directory.
- **Write (`w`)**: Needed to create new files or modify contents in a directory.
- **Execute (`x`)**: Needed to traverse directories and interact with their contents.