1. MODIFY any SOFT link and observer original?

- First, we make a folder called link and then create two text files inside it, namely, 1.txt 2.txt using the touch command.

```
shaji@MSI:~$ mkdir link
shaji@MSI:~$ cd
shaji@MSI:~$ cd link
shaji@MSI:~/link$ ls
shaji@MSI:~/link$ touch 1.txt 2.txt
```

- Now, create a symbolic or soft link using the following command.

```
shaji@MSI:~/link$ ln -s 1.txt 3.txt
shaji@MSI:~/link$ tree
.
├── 1.txt
├── 2.txt
└── 3.txt -> 1.txt
```

- We can see the inode number as well to confirm

```
shaji@MSI:~/link$ ls -i
 1812 1.txt  11701 2.txt  12849 3.txt
```

- When we modify the linked file(3.txt) we can see the modification reflected in in the original file (1.txt)

```
shaji@MSI:~/link$ cat > 3.txt
Shaji is my name
```

```
shaji@MSI:~/link$ cat 1.txt
Shaji is my name
```

2. MODIFY original file of SOFT link and observe?

- When I modified the original file the soft link also reflected the same changes

```
shaji@MSI:~/link$ nano 1.txt
shaji@MSI:~/link$ cat 1.txt
My number is 9848394238
shaji@MSI:~/link$ cat 3.txt
My number is 9848394238
```

1. **Remove any SOFT link and observer original?**

- The removal of the soft link didn't change anything in the original file.

```
shaji@MSI:~/link$ ls
1.txt  2.txt  3.txt
shaji@MSI:~/link$ rm 3.txt
shaji@MSI:~/link$ ls
1.txt  2.txt
shaji@MSI:~/link$ cat 1.txt
My number is 9848394238
shaji@MSI:~/link$ tree
.
├── 1.txt
└── 2.txt
```

2. **Remove original file of SOFT link and observe?**

- The soft link becomes inaccessible when we delete the original.

```
shaji@MSI:~/link$ ln -s 1.txt 3.txt
shaji@MSI:~/link$ tree
.
├── 1.txt
├── 2.txt
└── 3.txt -> 1.txt

0 directories, 3 files
shaji@MSI:~/link$ rm 1.txt
shaji@MSI:~/link$ tree
.
├── 2.txt
└── 3.txt -> 1.txt
```

1. **Modify any hard link and observe original**

- First, we created a hard link between two files file1.txt and hardlink.txt

```
shaji@MSI:~$ mkdir hardlink
shaji@MSI:~$ cd hardlink/
shaji@MSI:~/hardlink$ ls
shaji@MSI:~/hardlink$ touch file1.txt file2.txt
shaji@MSI:~/hardlink$ ls -li
total 0
17718 -rw-r--r-- 1 shaji shaji 0 Apr  3 12:58 file1.txt
77593 -rw-r--r-- 1 shaji shaji 0 Apr  3 12:58 file2.txt
shaji@MSI:~/hardlink$ ln file1.txt file2.txt
ln: failed to create hard link 'file2.txt': File exists
shaji@MSI:~/hardlink$ ln file1.txt hardlink.txt
shaji@MSI:~/hardlink$ ls -li
total 0
17718 -rw-r--r-- 2 shaji shaji 0 Apr  3 12:58 file1.txt
77593 -rw-r--r-- 1 shaji shaji 0 Apr  3 12:58 file2.txt
17718 -rw-r--r-- 2 shaji shaji 0 Apr  3 12:58 hardlink.txt
```

```
shaji@MSI:~/hardlink$ nano hardlink.txt
shaji@MSI:~/hardlink$ cat file1.txt
Hi

shaji@MSI:~/hardlink$ cat hardlink.txt
Hi
```

- When we modified anything in the hardlink.txt it reflected in the original file.

2. Modify original file of hard link and observe

- When we modified the original file1.txt, it reflected the modifications in the hardlink.txt

```
shaji@MSI:~/hardlink$ nano file1.txt
shaji@MSI:~/hardlink$ cat file1.txt
Hi, my name is Shajiuddin

shaji@MSI:~/hardlink$ cat hardlink.txt
Hi, my name is Shajiuddin
```

1. Remove any hard link and observe original

- There was no impact on the original file1.txt

```
shaji@MSI:~/hardlink$ tree
.
├── file1.txt
├── file2.txt
└── hardlink.txt

0 directories, 3 files
shaji@MSI:~/hardlink$ rm hardlink.txt
shaji@MSI:~/hardlink$ tree
.
├── file1.txt
└── file2.txt

0 directories, 2 files
shaji@MSI:~/hardlink$ cat file1.txt
Hi, my name is Shajiuddin
```

2. Remove original file of hard link and observe

- The linked file is still accessible with no impact on the linked file.

```
shaji@MSI:~/hardlink$ ls -i
17718 1.txt  77593 2.txt  81478 3.txt  17718 4.txt
shaji@MSI:~/hardlink$ rm 1.txt
shaji@MSI:~/hardlink$ ls -i
77593 2.txt  81478 3.txt  17718 4.txt
```

1. Comparison among soft & hard copy

- Soft Links are invalid once the source or parent file is deleted.

- Soft link is just a link that points to the original file yet with different inode numbers and permissions.
- The soft link is like a shortcut to a file. If you remove the file, the shortcut is useless.
- If we remove the soft link, the original file will still be available.
- Hard Link and source file have the same inode number and permissions.
- If we change the permissions on the source file, the same permission will be applied to the hard link file as well.
- If we change the permissions on the source file, the same permission will be applied to the hard link file as well.

Differences between “wget” & “curl” commands

- wget and curl are both command-line utilities used to download files and web pages from the internet. However, they have some key differences in terms of their features and functionality. Here are some of the main differences:
- wget can recursively download files, which means it can follow links within a web page and download all linked files. curl cannot do this by default, although it can be scripted to simulate this behaviour.
- curl supports many more protocols than wget, including FTP, SMTP, POP3, and more. wget primarily supports HTTP and HTTPS.
- curl is more flexible and customizable than wget. It supports many more options and can be used for more complex tasks, such as sending data to a web server using different methods (e.g. POST, PUT, DELETE).
- wget is installed by default on most Unix-based systems, while curl may need to be installed separately.
- In summary, wget is a simple and reliable tool for downloading files from the web, while curl is a more powerful and flexible tool that can be used for a wider range of tasks.

SCRIPTS

1. Write a script to print the date and redirect it to output.txt?

```
  GNU nano 6.2                    date_output.sh
date > output.txt
```

```
shaji@MSI:~$ nano
shaji@MSI:~$ nano date_output.sh
shaji@MSI:~$ chmod 777 date_output.sh
shaji@MSI:~$ ./date_output.sh
shaji@MSI:~$ ls
date_output.sh  hardlink  link  output.txt
shaji@MSI:~$ cat output.txt
Tue Apr  4 03:46:46 IST 2023
```

2. **Make a folder using the date as the name**

```
 GNU nano 6.2                    date_folder.sh
mkdir $(date '+%d-%m-%Y')
```

```
shaji@MSI:~$ nano date_output.sh
shaji@MSI:~$ nano date_folder.sh
shaji@MSI:~$ chmod 777 date_folder.sh
shaji@MSI:~$ ./date_folder.sh
shaji@MSI:~$ ls
04-04-2023      date_output.sh  link
date_folder.sh  hardlink        output.txt
```

3. i) Create a bash script to print the local time, date, username of your system, and your current path.
   ii) After printing, redirect the output into a file called output.txt
   iii) Insert output.txt into a new directory, where the directory name is the current timestamp.

```
# Get current date and time in the desired format
timestamp=$(date +%Y-%m-%d_%H-%M-%S)

# Get current user and path
user=$(whoami)
path=$(pwd)

# Print information to console
echo "Local Time: $(date +%T)"
echo "Date: $(date +%Y-%m-%d)"
echo "Username: $user"
echo "Current Path: $path"

# Redirect output to file
echo -e "\nOutput saved to output.txt"
echo -e "Local Time: $(date +%T)\nDate: $(date +%Y-%m-%d)\nUsername: $user\nCurrent Path: $path" > output.txt

# Create directory and move output file into it
mkdir "$timestamp"
mv output.txt "$timestamp"
echo -e "\nOutput directory created: $timestamp"
```

```
shaji@MSI:~$ nano date_folder.sh
shaji@MSI:~$ nano info_print.sh
shaji@MSI:~$ chmod 777 info_print.sh
shaji@MSI:~$ ./info_print.sh
Local Time: 04:00:36
Date: 2023-04-04
Username: shaji
Current Path: /home/shaji

Output saved to output.txt

Output directory created: 2023-04-04_04-00-36
shaji@MSI:~$ ls
04-04-2023            date_folder.sh  hardlink       link
2023-04-04_04-00-36   date_output.sh  info_print.sh
shaji@MSI:~$ cd 2023-04-04_04-00-36/
shaji@MSI:~/2023-04-04_04-00-36$ ls
output.txt
shaji@MSI:~/2023-04-04_04-00-36$ cat output.txt
Local Time: 04:00:36
Date: 2023-04-04
Username: shaji
Current Path: /home/shaji
```

4. **Create a bash script to execute the date every 2 minutes once on Saturdays only**

```
  GNU nano 6.2                    saturday.sh
echo $(date) > output.log
```

```
*/2 * * * 6  /home/shaji/saturday.sh
```

5. **Take a backup of a folder every month twice**

```
  GNU nano 6.2                  backup_script.sh
cp -r  ~/logs ~/backup
```

```
shaji@MSI:~$ ./backup_script.sh
shaji@MSI:~$ ls
04-04-2023            backup_script.sh  hardlink      logs
2023-04-04_04-00-36   date_folder.sh    info_print.sh output.log
backup                date_output.sh    link          saturday.sh
```

```
shaji@MSI:~$ cd logs
shaji@MSI:~/logs$ ls
1.txt  2.txt  3.txt
shaji@MSI:~/logs$ cd
shaji@MSI:~$ cd backup
shaji@MSI:~/backup$ ls
logs
shaji@MSI:~/backup$ cd logs/
shaji@MSI:~/backup/logs$ ls
1.txt  2.txt  3.txt
shaji@MSI:~/backup/logs$
```

```
0 0 1,15 * * /home/shaji/backup_script.sh
```

6. **Print the count of the number of files in a folder & print the output in a file called count.txt**

```
  GNU nano 6.2                    count_files.sh
ls -1 ~/ | wc -l > count.txt
```

```
shaji@MSI:~$ nano count_files.sh
shaji@MSI:~$ chmod 777 count_files.sh
shaji@MSI:~$ ./count_files.sh
shaji@MSI:~$ ls
04-04-2023            count.txt         hardlink      output.log
2023-04-04_04-00-36   count_files.sh    info_print.sh saturday.sh
backup                date_folder.sh    link
backup_script.sh      date_output.sh    logs
shaji@MSI:~$ cat count.txt
14
```

7. **Create files dynamically everyday at 12 AM where the file name is a date**

```
filename=$(date +%d-%m-%Y)

touch ~/$filename.txt
```

```
shaji@MSI:~$ ./weekly_date.sh
shaji@MSI:~$ ls
04-04-2023              backup_script.sh
04-04-2023.txt          count.txt
```

```
0 0 * * * /home/shaji/weekly_date.sh
```

Crontab scripts labs

1. **Write a script to print the current directory and username and redirect it to a file called output.txt**

```
  GNU nano 6.2                          print_info.sh
(echo "Current directory: $(pwd)"
echo "Username: $(whoami)"
echo "Current date: $(date)") > output.txt
```

```
shaji@MSI:~/scripts$ nano print_info.sh
shaji@MSI:~/scripts$ chmod 777 print_info.sh
shaji@MSI:~/scripts$ ./print_info.sh
shaji@MSI:~/scripts$ ls
output.txt  print_info.sh
shaji@MSI:~/scripts$ cat output.txt
Current directory: /home/shaji/scripts
Username: shaji
Current date: Tue Apr  4 17:01:49 IST 2023
```

2. **Create a file with the current timestamp as its name inside a folder with the current date as its name?**

```
folder_name=$(date +%d-%m-%Y)
mkdir -p $folder_name

file_name=$(date +%d-%m-%Y_%H-%M-%S).txt
touch $folder_name/$file_name
```

```
shaji@MSI:~/scripts$ nano print_info.sh
shaji@MSI:~/scripts$ nano date_folder.sh
shaji@MSI:~/scripts$ chmod 777 date_folder.sh
shaji@MSI:~/scripts$ ./date_folder.sh
shaji@MSI:~/scripts$ ls
04-04-2023  date_folder.sh  output.txt  print_info.sh
```

```
shaji@MSI:~/scripts/04-04-2023$ ls
04-04-2023_17-08-05.txt
```

3. **Create a bash script to print the local time, date, username of your system, and your current path and redirect the output into a file called output.txt. Insert output.txt into a new directory, where the directory name is the current timestamp.**

```
  GNU nano 6.2                    more_info.sh *
folder_name=$(date +%d-%m-%Y_%H:%M:%S)
mkdir -p $folder_name

(echo "Local time: $(date +%T)"
echo "Local date: $(date +%F)"
echo "Username: $(whoami)"
echo "Current path: $(pwd)") > output.txt

mv output.txt $folder_name/
```

```
shaji@MSI:~/scripts$ nano more_info.sh
shaji@MSI:~/scripts$ chmod 777 more_info.sh
shaji@MSI:~/scripts$ ./more_info.sh
shaji@MSI:~/scripts$ ls
04-04-2023            date_folder.sh  print_info.sh
04-04-2023_17:19:20   more_info.sh
shaji@MSI:~/scripts$ cd 04-04-2023_17\:19\:20/
shaji@MSI:~/scripts/04-04-2023_17:19:20$ ls
output.txt
shaji@MSI:~/scripts/04-04-2023_17:19:20$ cat output.txt
Local time: 17:19:20
Local date: 2023-04-04
Username: shaji
Current path: /home/shaji/scripts
```

4. **Write a script to print the count of the number of files in a folder and redirect the count to a file called count.txt**

```
  GNU nano 6.2                    count_files.sh
ls -1 ~/ | wc -l > count.txt
```

```
04-04-2023            count.txt       info_print.sh  scripts
04-04-2023.txt        count_files.sh  link           weekly_date.sh
2023-04-04_04-00-36   date_folder.sh  logs
backup                date_output.sh  output.log
backup_script.sh      hardlink        saturday.sh
shaji@MSI:~$ nano count_files.sh
shaji@MSI:~$ ./count_files.sh
shaji@MSI:~$ cat count.txt
17
```

5. **Create a bash script to execute the date every 2 minutes once on weekends only**

```
  GNU nano 6.2                    weekend_date.sh
date > weekend.txt
```

```
*/2 * * * 6-7 /home/shaji/weekend_date.sh
```

6. **Take a backup of a folder daily twice**

```
  GNU nano 6.2                  backup_script.sh
cp -r  ~/logs ~/backup
```

```
0 1,15 * * * /home/shaji/backup_script.sh
```