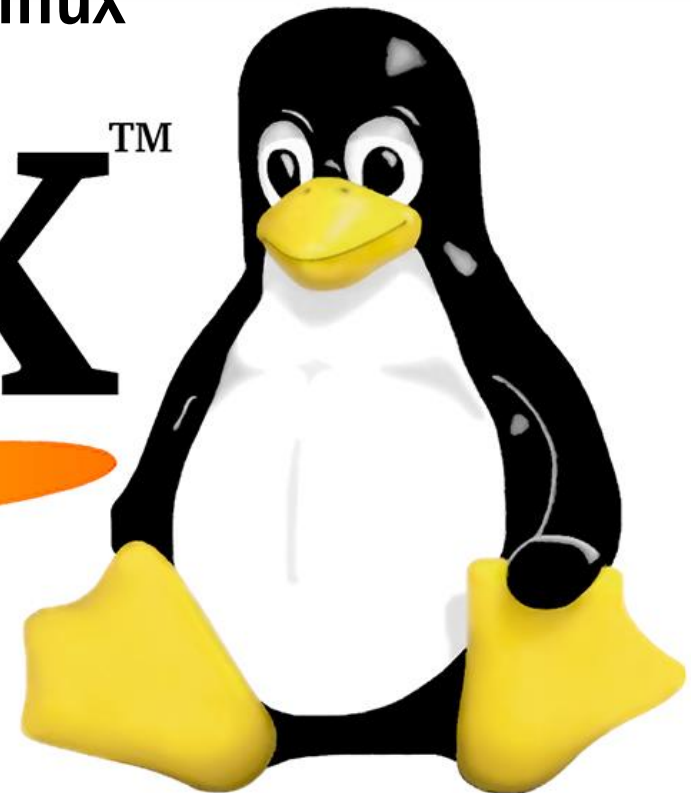


# Lab3

## File Systems (I) (Structure and File Types)

Comp311- Lab Linux

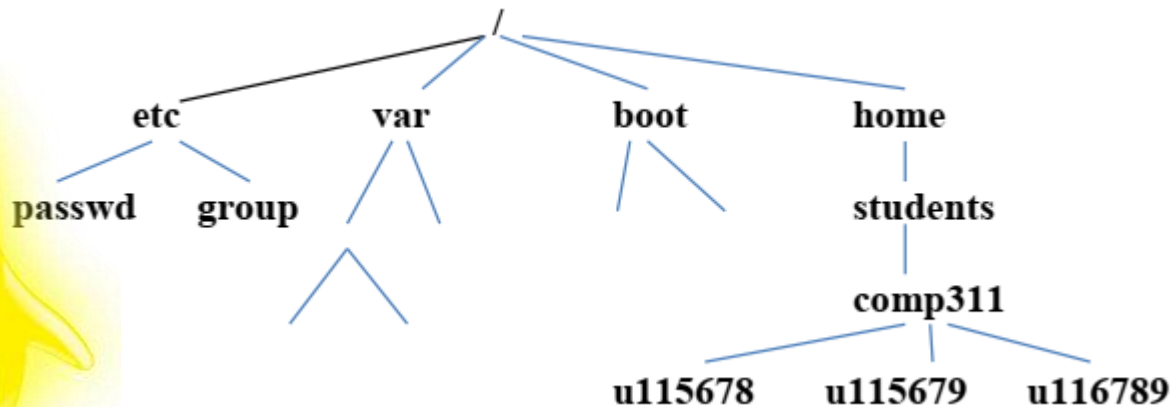
# Linux<sup>TM</sup>



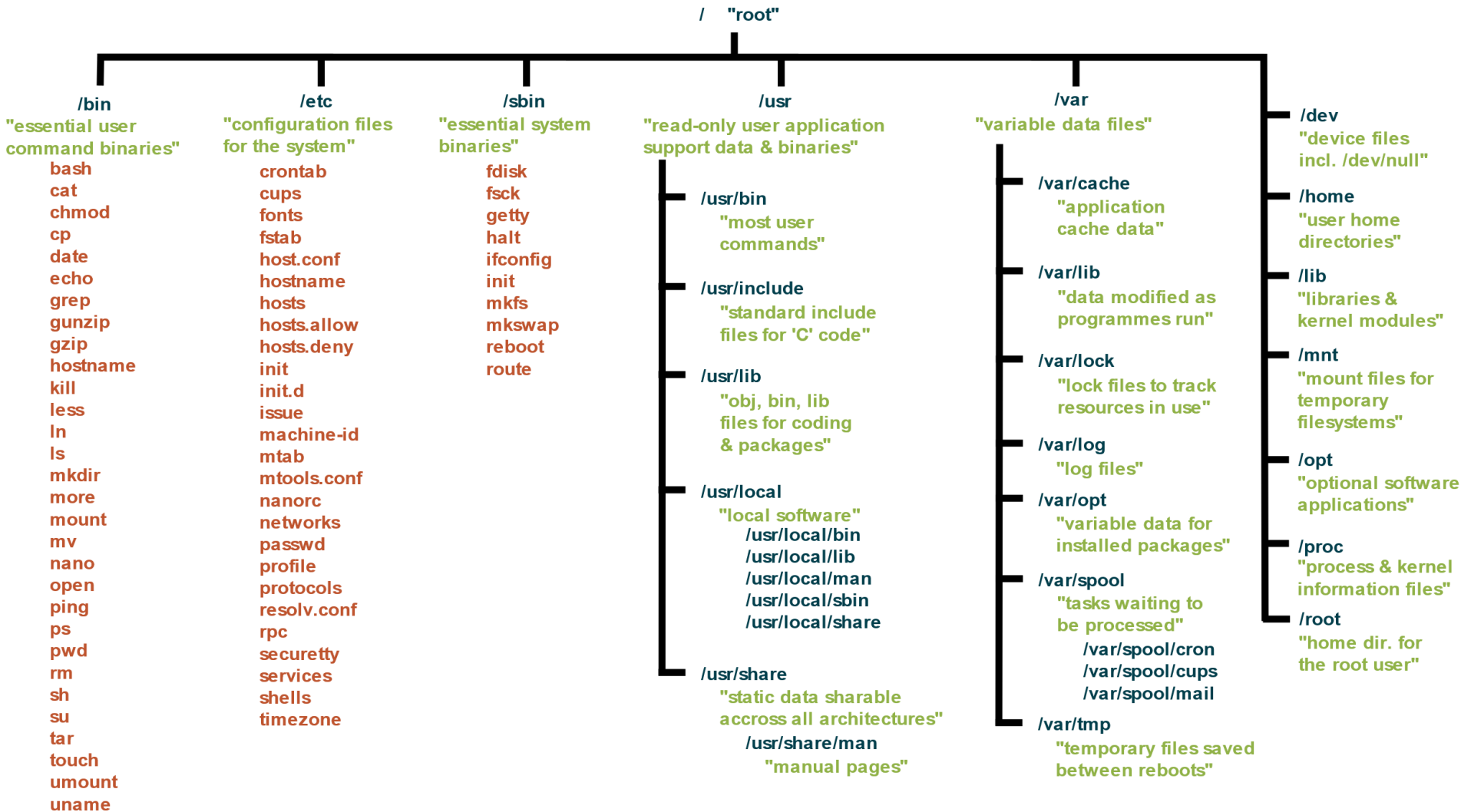
Instructor :Murad Njoum

# Objectives

- After completing this lab, the student should be able to:
- Understand the structure of the Linux file system
  - Build tree structures using **absolute and relative paths**
  - Recognize and create the different main Linux file types



# Linux File Systems Structure



# Linux File Systems

To display the file systems, use the **command *df*** (display file systems) as follows:

## Run Command

1. *df -h*

2. *df -h T*



*df -h* ( -h human readable format). (-T type ,ext2, ext4, nfs4, btrfs, xfs)

# Linux File Systems



Feature	EXT4	XFS	BTRFS
Architecture	Hashed B-tree	B+ tree	Extent based
Introduced	2006	1994	2009
Max volume size	1 Ebytes	8 Ebytes	16 Ebytes
Max file size	16 Tbytes	8 Ebytes	16 Ebytes

Feature	4 bit	FAT	FAT32	exFAT	NTFS	ReFS	
Max number of files	4 bit						
Max file name size	255	Maximum volume size	4 GB	32 GB	128 PB	256 TB	4.7 ZB (zettabytes)
Attributes	Yes	Maximum file size	4 GB	4 GB	16 EB (exabytes)	18 EB (exabytes)	18 EB (exabytes)
Transparent compression	No	Maximum filename length	8.3 characters	255 characters	255 characters	255 characters	255 characters
Transparent encryption	Yes	Maximum cluster size	64 KB	32 KB	32 MB	2048 KB	64 KB
Copy-on-Write (COW)	No	File compression	No	No	No	Yes	No
		File encryption	No	No	No	Yes	No
Snapshots	No	Permissions	No	No	No	Yes	Yes

# Linux File Systems

To Manipulate directories under a file system, you can use the following commands:

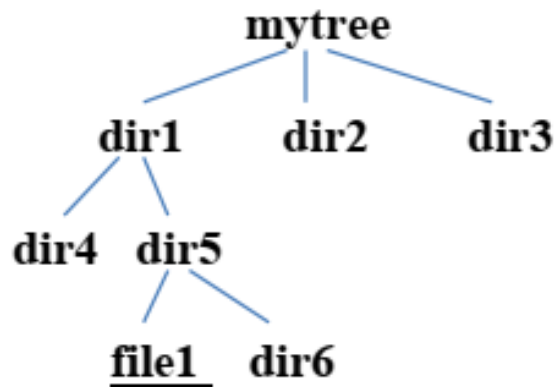
	Command	Explanation
1	<i>mkdir newdir</i>	creates a new directory called newdir
2	<i>cd newdir</i>	changes your position to newdir
3	<i>rmdir newdir</i>	removes directory new directory only if newdir is empty
4	<i>rm -rf newdir</i>	removes non-empty or empty directory newdir
5	<i>pwd</i>	displays present working directory



# Practice.1:



Using the commands above, create the following tree under your home directory:



To display your tree use the command:  
**ls -R mytree**

## Direct thinking :

```
mkdir mytree
cd mytree
mkdir dir1 dir2 dir3
cd dir1
mkdir dir4 dir5
cd dir5
mkdir dir6
touch file1
```

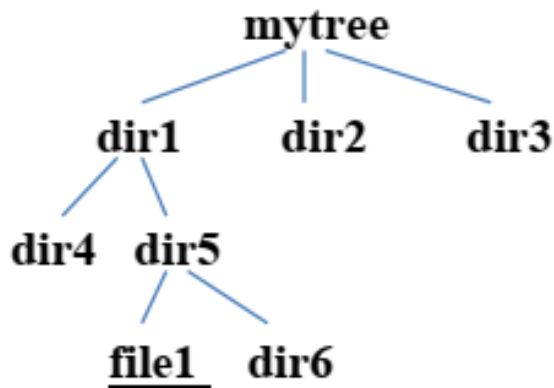
# Practice.2-1:



A. Delete tree structure you created at your home directory

**rm -rf mytree? why**

B. Rebuild the same tree at you home directory.



```
mkdir mytree
```

```
mkdir mytree/dir1 mytree/dir2 mytree/dir3
```

```
mkdir mytree/dir1/dir5
```

```
mkdir mytree/dir1/dir5/dir6
```

....

```
rm -rf mytree
```

Try:

```
mkdir mytree/dir1 mytree/dir2 mytree/dir3
```

```
mkdir -p mytree/dir1 mytree/dir2 mytree/dir3
```

```
mkdir -p mytree/{dir2,dir3,dir1/{dir4,dir5/dir6}};touch mytree/dir1/dir5/file1
```



# Practice.2-2:

A. Delete tree structure you created at your home directory

```
rm -rf mytree
```

B. Construct The previous tree (mytree) using absolute path method in one line ?

```
mkdir -p mytree/{dir2,dir3,dir1/{dir4,dir5/dir6}};touch mytree/dir1/dir5/file1
```



# Absolute and Relative Paths

Any Linux file may be referenced by either its **absolute path name** or **relative path name**. Each file **has one and only one absolute path name** while it may have an **infinite number of relative names**.

The absolute path name (**from root, /, to location of file**) **for file1** in the previous tree is:

***/home/students/comp311/username/mytree/dir1/dir5/file1***

While it has several relative names such as:

***./mytree/dir1/dir5/file1***

***./mytree/../../mytree/dir1/dir4/../../dir5/file1***

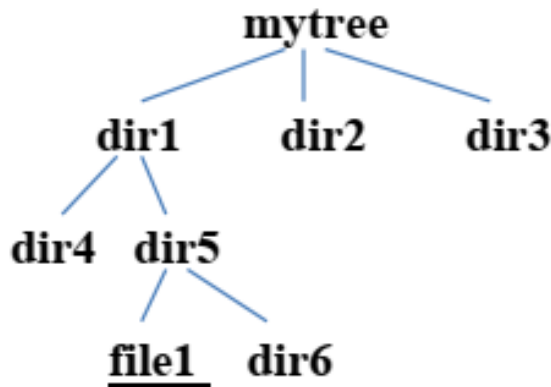
and so forth. **Notice that ( . ) stands for current directory while ( .. ) stands for previous (parent) directory.**

# Practice.3:

A. Remove the sub tree *mytree* created in the previous section ( `rm -rf mytree` )

Now try creating the whole tree again using relative paths from your home directory (i.e.

you are not allowed to use the **cd** command to create any parts of the tree)



```
mkdir mytree  
mkdir mytree/dir1  
mkdir mytree/dir2  
mkdir mytree/dir3
```

```
mkdir mytree/dir1/dir4  
mkdir mytree/dir1/dir5  
mkdir mytree/dir1/dir5/dir6  
touch mytree/dir1/dir5/dir6/file1
```



B. Construct The previous tree (mytree) using relative path method in one line ?

```
mkdir -p mytree/dir1/./dir2/./dir3/./dir1/dir4/./dir5/dir6
```

who wants  
to win...

**What is the difference between ?**

**ls /mytree/dir1**

**ls mytree/dir1**

**ls ./mytree/dir1**





**ls /mytree: from root**

**ls mytree**

**ls ./mytree :from current position**

So far, we have mentioned two types of Linux files as follows:

1- **Regular files** which include **scripts, binaries**, as well as **text files**. These files contain data and are identified by any empty first slot when we list them using the ***ls -al filename*** command. These are created using the ***vi*** editor.

2- **Directories** which are simply containers that **include the mappings** between filenames and subdirectory names and their unique inode (index) number in the file system. These are identified by the letter ***d*** in the first slot when we list them using the ***ls -al dirname*** command. These are created using the ***mkdir*** command.

```
drwxrwxr-x 2 mnjourn mnjourn 4096 Sep 23 18:38 dir4
```

```
-rw-rw-r-- 1 mnjourn mnjourn 0 Sep 23 19:14 file1
```

## The third type of Linux files are the special (device) :

3. **The third type** of Linux files are the **special (device)** files which are usually located under the **/dev** directory. There are two types of device files:

1- **Character device files**: which are used to read and write from/to devices **one character** at a time (e.g. **keyboard device files**). These are identified with the character **c** when we list them with the `ls -al` command.

2- **Block device files**: which are used to read and write from/to devices **one block** at a time (e.g. disk device files). ). These are identified with the character **b** when we list them with the `ls -al` command.

```
brw-rw---- 1 root disk 8, 1 Sep 23 16:11 sda1
```

```
crw--w---- 1 root tty 4, 19 Sep 23 16:11 tty19
```

**Run the command `ls -al` on the `/dev` directory. List three character device files and three block device files.?**

# The fourth type of Linux files are the links:

The original links in Linux are called the **hard links** and are created using the command

*In.*

## Practice:

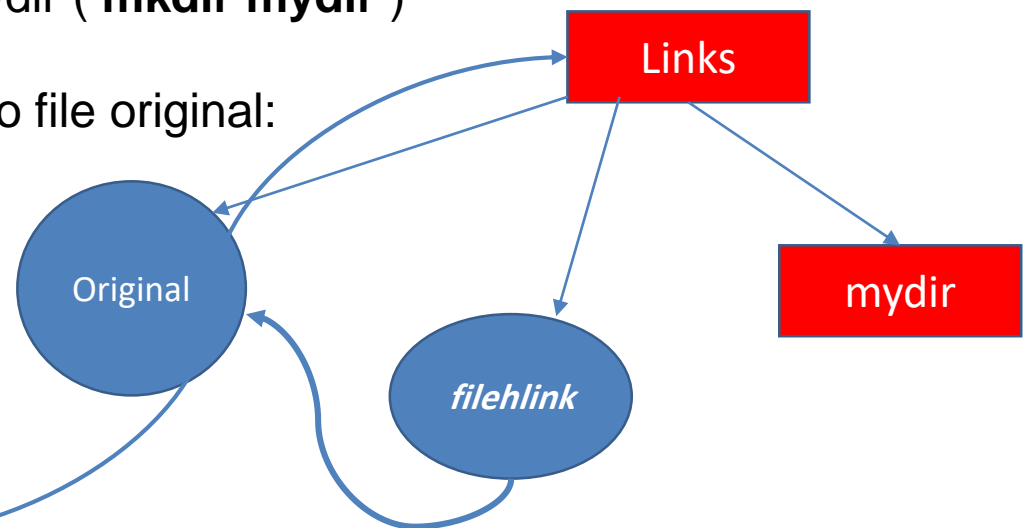
1. Create a directory to try some links ( **mkdir links; cd links** )

Create a file called original and ( *vi original* ) put the phrase “**this is original**” inside then save and quit

2. Now create a directory called mydir ( **mkdir mydir** )

Let us now start working with links:

Create a hard link called **filehlink** to file original:



***From inside directory links  
run command***

*ln original filehlink*



# The fourth type of Linux files are the links:

List the files in your directory with *ls -ali* ( the i option displays the inode numbers)  
Notice that all the properties of file original and link filehlink (**except the name**) are exactly the same (**even the inode number**). Hard links basically give a new name to the same inode number.

```
19531008 -rw-rw-r-- 2 mnjourn mnjourn 26 Sep 23 19:59 filehlink
```

```
19531008 -rw-rw-r-- 2 mnjourn mnjourn 26 Sep 23 19:59 orignal
```

## Hard links have two limitations:

1- Not allowed on directories

Try the command:

*In mydir dirhlink*

*What was the result?*

*hard link not allowed for directory*

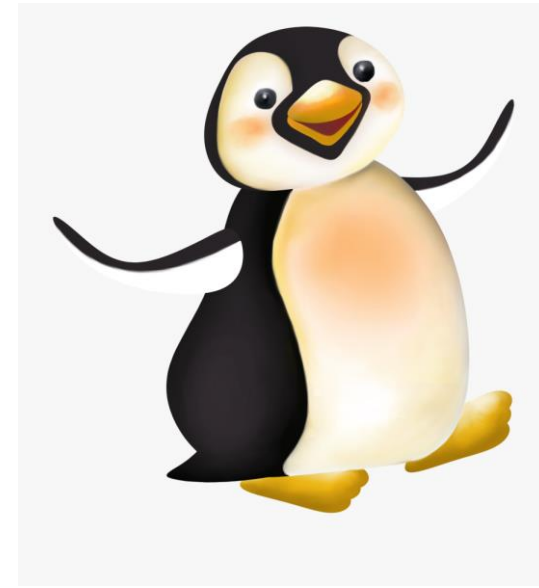
2- Not allowed across different devices (file systems)

Try the command:

*In /etc/passwd passwdhlink*

*What was the result?*

*Invalid cross-device link*



# symbolic (soft)

Those limitations are solved using a different type of links called symbolic (soft) links.

To create a symbolic link simply use the option (-s) with the *ln* command.

Try the following commands and see what happens:

*rm filelink*

*ln -s original fileslink*

*ls -ali* (what are the differences between original and fileslink properties?)

```
mnjourn@ubuntu:~/links$ ls -ali
total 76
19531017 drwxr-xr-x  5 mnjourn mnjourn  4096 Sep 23 20:38 .
19136520 drwx----- 72 mnjourn mnjourn 32768 Sep 23 19:59 ..
19531018 drwxr-xr-x  2 mnjourn mnjourn  4096 Feb 28  2018 dir1
19531019 drwxr-xr-x  2 mnjourn mnjourn  4096 Feb 28  2018 dir2
19531007 lrwxrwxrwx   1 mnjourn mnjourn    5 Sep 23 20:26 dirhlink -> mydir
19531010 lrwxrwxrwx   1 mnjourn mnjourn    7 Sep 23 20:38 fileslink -> original
21890415 drwxrwxr-x   2 mnjourn mnjourn  4096 Sep 23 20:00 mydir
19531008 -rw-rw-r--   1 mnjourn mnjourn   26 Sep 23 19:59 original
19531009 lrwxrwxrwx   1 mnjourn mnjourn   11 Sep 23 20:28 passwdhlink -> /etc/passwd
```

# Practice.4:

**ls -ali** (what are the differences between **original** and **filelink** properties?)

---

**ln -s mydir dirlink** (what happened now?)

---

**ln -s /etc/passwd passwdlink** (what happened now?)

---

# who wants to win...

- 1. What will be happen with hard link if we delete the original file ?**
- 2. What will be happen if we delete soft (symbolic) link ?**





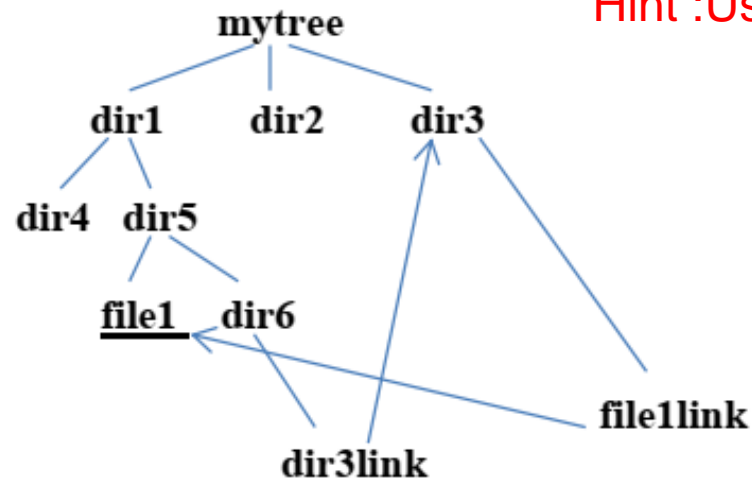
# 1. What will be happen with hard link if we delete the original file ?

**Hard link** : Original deleted, link not affected, still can edit it (as a copy of original file)

# 2. What will be happen if we delete soft (symbolic) link ?

**Soft link** : Original deleted, link destroyed, can edit it the original (empty file)

Now go back to the tree you created earlier (*mytree*) and add the links shown below:



Hint :Using relative path methods!!

What commands did you use:

1. `cd mytree/dir3`  
`In ../../dir1/dir5/file1 file1link`
2. `cd mytree/dir1/dir5/dir6`  
`In -s ../../../../dir3 dir3link`

Although there are few more types of Linux files such as pipe files or socket files. They are rarely used or seen by users.

**Another solution : `In -srf mytree/dir1/dir5/file1 mytree/dir3`**



*Thank You Guys*

**Published By: Murad Njoum**