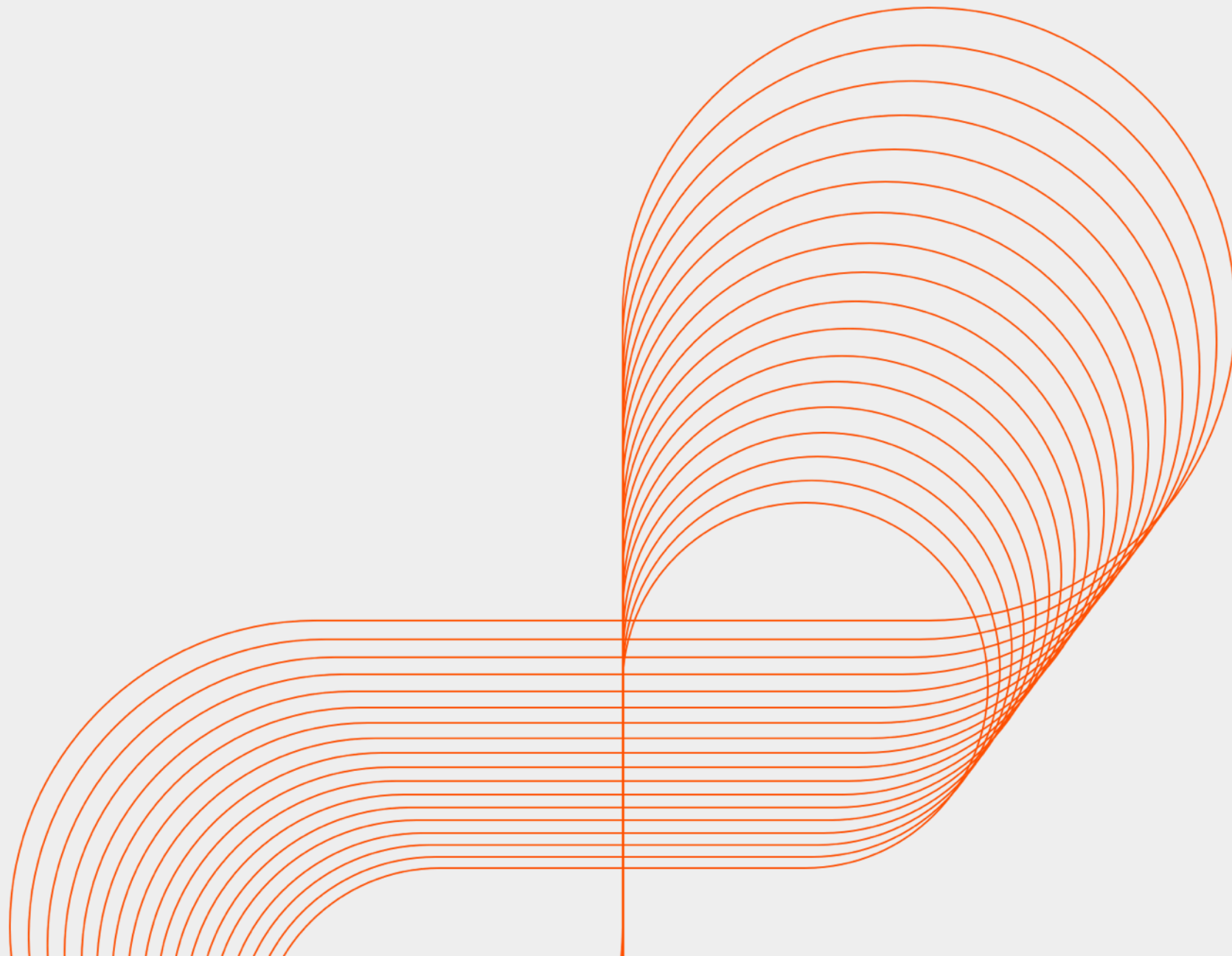




Persistent

UNIX



Files & Directories

A decorative graphic consisting of a horizontal orange line that extends across the width of the slide. From the right end of this line, a vertical orange line descends, and a large orange circle is drawn, partially overlapping the horizontal line and the vertical line.

Unix Concept Of File

- In Unix, everything is considered to be a file, including physical devices such as DVD-ROMs, USB devices, etc
- This allows Unix to be consistent in its treatment of resources and gives user a consistent and device independent mechanism of interaction with the systems
- For example, printing a file on a printer is treated similarly to displaying it on the terminal screen

Unix File Types

- Unix can have the following types of files
 - **Regular files** - these can be text or binary files
 - **directory** - a special file that stores links to other file. Symbolized by d
 - **link** - a file that references to other file. It is marked with l
 - **named pipe** - special files that used for interprocess communication. Marked with a p
 - **socket** - special file used for interprocess communication. Marked with s
 - **device file** - files that represent the hardware in the system. It has two types
 - character device, marked as c & block device, marked as b

stat command

- stat command will display file or file system status.

example – stat filename

stat command will display the following file system status like.

File Name, Size, Block, Filetype, Links,

File Access permission (0664/ -rw-rw-r--)

Uid, Gid

File access time, modify time, change time etc.

File System Basics

- A file system is a logical collection of files on a partition or a disk
- In Unix, it is normal to have more than one partition. Usually, `/`, `/home` are `/temp` mounted on different partitions
- This helps in logical maintenance & management of differing file systems. These partitions are invisible to users, who can move from one filesystem to other without knowing it

Advantages Of Partitioning

- **Ease of use** - Make it easier to recover a corrupted file system or operating system installation.
- **Performance** - Smaller file systems are more efficient. You can tune file system as per application such as log or cache files. Dedicated swap partition can also improve the performance
- **Security** - Separation of the operating system files from user files may result into a better and secure system. Restrict the growth of certain file systems is possible using various techniques.
- **Backup and Recovery** - Easier backup and recovery.
- **Stability and efficiency** - You can increase disk space efficiency by formatting disk with various block sizes. It depends upon usage. For example, if the data is lots of small files, it is better to use small block size.
- **Testing** - Boot multiple operating systems such as Linux, Windows and FreeBSD from a single hard disk.

Partition Naming In Linux

- Linux refers to disk partitions using a combination of letters and numbers in the form

`/dev/xxyN`

- `/dev` - this string is the name of the directory in which all device files reside
- `xx` - the first two letters indicate the type of device on which the partition resides. It is `hd` for IDE disks and `sd` for SCSI disks
- `y` - indicates which device the partition is on
- `N` - denotes the partition. 1 through 4 are primary or extended partitions. Logical partitions start at 5

Configuration Files

- The information about partitions and mount points is kept in /etc/fstab file (file system table)
- You can find how many partitions are on your disk(s) through
 - /proc/partitions file
 - /dev/sda* files in /dev directory

File System Structure

- Boot Block : information needs to boot the system
- Super Block : File System Specifications
 - Size
 - Max. number of files
 - Free blocks
 - Free inodes
- inode List
- Block List : The files data



Inode

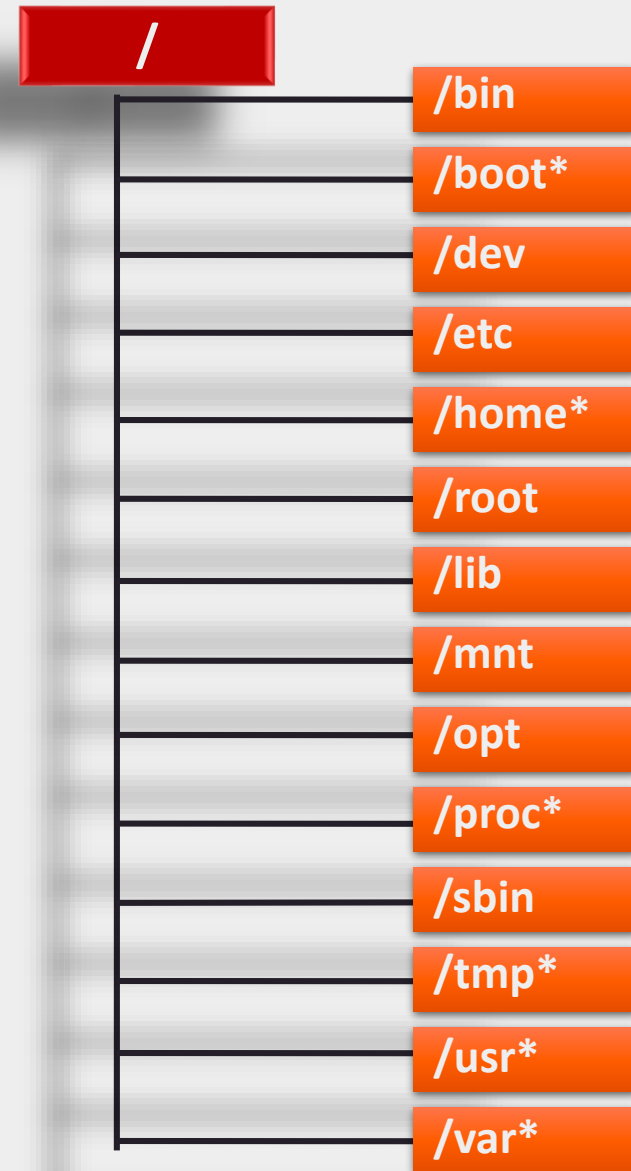
- In Unix, the internal representation of a file is given by inode, that contains a description of the disk layout of the file data and other information, such as the file owner, access permissions & times.
- Every file will have one inode but it can have several names (called as links) all of which map into the inode.
- When a process refers a file by name, the kernel parses the filename components, checks if the process has permission to search in the file directory & retrieves the inode of the file
- When a new file is created, the kernel assigns it an unused inode
- You can view the inodes of files through `ls -li` command

Unix File System Details

- Unix file system is based on Filesystem Hierarchy Standard (FHS)
- It defines the main directories and their contents in Unix based operating systems
- The FHS is maintained by the Free Standards Group, a non-profit organization consisting of major software and hardware vendors such as HP, IBM, Red Hat, etc
- But not all Unix variants follow the proposed standard completely

Directory Structure

- The directories noted here are a subset of those specified by FHS
- Everything starts with the root directory, usually designated only by `/`. The file system uses a hierarchical file structure and all the other directories originate from here
- Every Unix system will have most of the directories shown on the right, although not every version will have all the directories and some may even incorporate their own



Directories & Paths

Symbol

Meaning

| | | |
|----|--|---|
| . | Stands for the current directory | |
| .. | Stands for the next directory up the | tree (parent directory) |
| ~ | Home directory tool | |
| / | If used immediately after the space directory; otherwise, used in | that follows <i>cd</i> , directs to root designating paths |

Relative & Absolute Path

- Every file in the filesystem can be accessed using an absolute path & a relative path.
- The absolute path means the exact location of the file in its file system while relative path refers to the location of the file in relation to your current directory .e.g. if you are in */etc* directory, the relative path to */etc/passwd* is *passwd* and the absolute path is */etc/passwd*
- Unix is a case-sensitive OS. If you have two files named *account.doc* and *Account.doc*, both are different (as opposed to same on Windows)

FileSystem Commands

| <i>Command</i> | Function |
|-----------------------|---|
| <i>cd</i> | Change directory (if run by itself, takes a user to his home directory) |
| <i>mkdir</i> | Make directory |
| <i>cp</i> | Copy files |
| <i>rm</i> | Remove files |
| <i>rmdir</i> | Remove directory |
| <i>mv</i> | Move files |
| <i>find</i> | Find files |
| <i>du</i> | Gives the estimate of space used by files/directories |
| <i>df</i> | Estimates the space used by a file system |

Assignments

- Move a directory (with all files) to another location
- Remove all the files in a directory & then delete the directory
- Find all the C++ (.cpp) files in the home directory and all directories under it.
- Check how much space your home directory is using. Display in kilo bytes
- Display the total capacity of the disk drive along with the free & used space

File & Directory Permissions

- Since, Unix is a multi-user environment, security of user and system data is very important
- Users are given restricted access so that they don't have access to other users' files
- To view the permissions of the files and directories, use the **ls -l** command

```
[sid@vmrhel4 ~]$ ls -l
total 64
-rw-r--r--  1 sid accounts  20 Mar  8 06:29 company.info
-rw-r--r--  1 sid accounts  23 Mar  8 06:27 expenses
-rw-r--r--  1 sid accounts  17 Mar  8 06:28 loans
-rw-r--r--  1 sid accounts  40 Mar  8 06:28 payments
-rw-r--r--  1 sid accounts  15 Mar  8 06:25 salaries.March
drwxr-xr-x  2 sid accounts 4096 Mar  8 06:32 sales
-rw-r--r--  1 sid accounts  12 Mar  8 06:28 vendors.Banglore
-rw-r--r--  1 sid accounts  12 Mar  8 06:28 vendors.Pune
-rw-r--r--  1 sid accounts  15 Mar  8 06:28 vendors.Pune
-rw-r--r--  1 sid accounts  15 Mar  8 06:28 vendors.Pune
-rw-r--r--  1 sid accounts  15 Mar  8 06:28 vendors.Pune
```

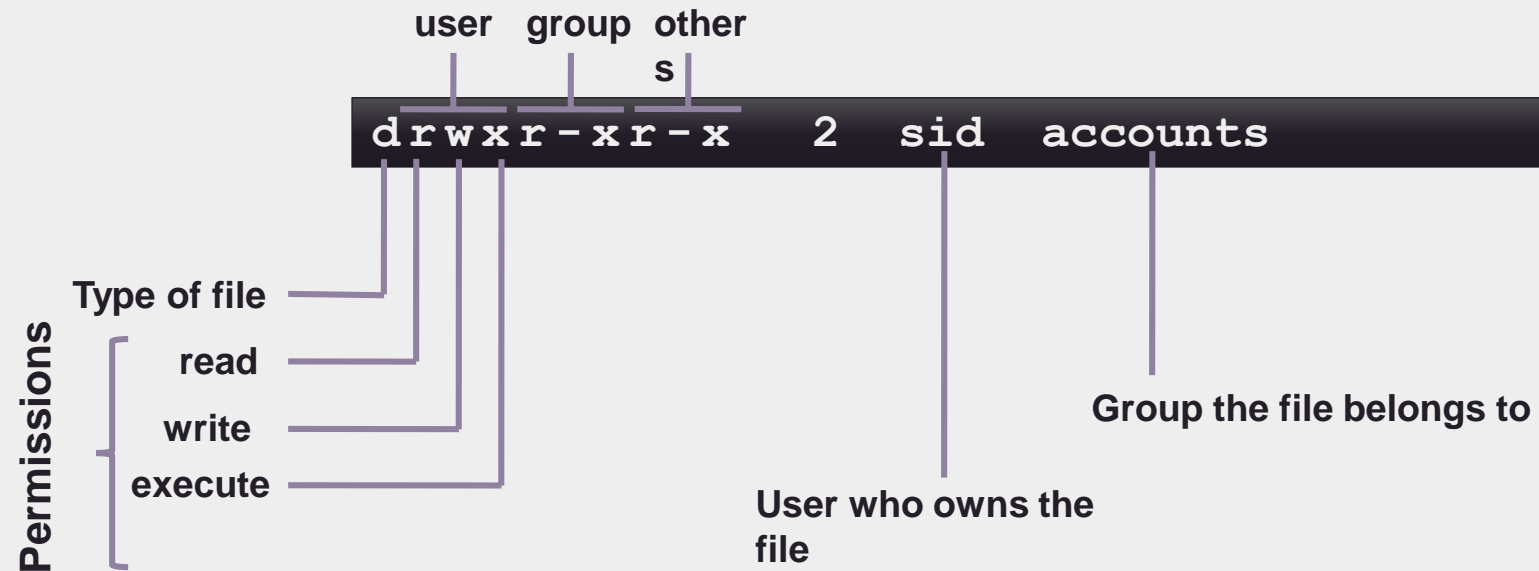
Understanding Permissions

- The three basic permissions are
 - **Read** – most basic operation required to open the file for reading. If applied to directory, it will allow you to list the contents of the directory, but not necessarily read the files it contains.
 - **Write** – allows you to write in a file. If applied to directory, you can create & remove files/directories
 - **Execute** – required to execute a file. It is never set by default, thus makes Linux immune to viruses. This permission is also required to execute(enter) a directory and is usually the default permission for every directory

Understanding Permissions

| Permission | Applied to Files | Applied to Directories |
|------------|---------------------------|------------------------------|
| Read | Open a file | List contents of a directory |
| Write | Change contents of a file | Create & delete files |
| Execute | Run a program file | Change to the directory |

Permissions



Setting Permissions

- Permissions can be changed through *chmod* command

chmod who(=+-)permissions filename

- The “*who*” is a list of letters that specifies whom you’re giving the permissions to. These may be specified in any order
- Permissions are the same letters that you see in the directory listing

| Letter | Meaning |
|--------|--|
| u | The user who owns the file |
| g | The group the file belongs to |
| o | The other users |
| a | all of the above (abbreviation for ugo) |

| Letter | Meaning |
|--------|------------------------------|
| r | read permission |
| w | write (or delete) permission |
| x | execute permission |

Relative Mode

- Changing permissions as shown in the previous slide is called the relative mode
- **+** : will add permissions while keeping the existing ones for that entity
- **-** : will remove permissions while keeping existing ones for that entity
- **=** : assigns permissions while removing the existing ones for that entity

Relative Mode

- When changing permissions in relative mode, you may omit the “whom” part to add or remove permissions for all entities e.g. `chmod +x <file>` will add execute permission for all users (same as `chmod a+x <file>` & `chmod ugo+x <file>`)
- You may assign permissions to multiple entities at the same time e.g. `chmod g+w, o-r <file>` would add write permission to the group & remove read for others

Changing File Permissions

```
[root@vmrhelu4 sales]# ls -l
total 16
-rw-r--r--  1 sid accounts 20 Mar  8 06:29 company.info
-rw-r--r--  1 sid accounts 15 Mar  8 06:25 salaries.March
[root@vmrhelu4 sales]# chmod go=-r salaries.March
[root@vmrhelu4 sales]# ls -l
total 16
-rw-r--r--  1 sid accounts 20 Mar  8 06:29 company.info
-rw-----  1 sid accounts 15 Mar  8 06:25 salaries.March
[root@vmrhelu4 sales]# chmod go+w company.info
[root@vmrhelu4 sales]# ls -l
total 16
-rw-rw-rw-  1 sid accounts 20 Mar  8 06:29 company.info
-rw-----  1 sid accounts 15 Mar  8 06:25 salaries.March
[root@vmrhelu4 sales]# chmod o-w company.info
[root@vmrhelu4 sales]# ls -l
total 16
-rw-rw-r--  1 sid accounts 20 Mar  8 06:29 company.info
-rw-----  1 sid accounts 15 Mar  8 06:25 salaries.March
[root@vmrhelu4 sales]# █

[rooŧ@vmlrhelu4 sals]# █
-lw-----  1 sid accounts 15 Mar  8 06:25 salaries.March
-lw-lw-l--  1 sid accounts 20 Mar  8 06:29 company.info
total 16
[rooŧ@vmlrhelu4 sals]# ls -l
[rooŧ@vmlrhelu4 sals]# chmod o-w company.info
-lw-----  1 sid accounts 15 Mar  8 06:25 salaries.March
```

Changing Directory Permissions

```
[root@vmrhelu4 sales]# ls -l
total 24
-rw-rw-r-- 1 sid  accounts  20 Mar  8 06:29 company.info
drwxr-xr-x 2 root root    4096 Mar  8 08:08 profiles
-rw----- 1 sid  accounts  15 Mar  8 06:25 salaries.March
[root@vmrhelu4 sales]# chmod o=wx profiles/
[root@vmrhelu4 sales]# su sid
[sid@vmrhelu4 sales]$ cd profiles/
[sid@vmrhelu4 profiles]$ ls
ls: .: Permission denied
[sid@vmrhelu4 profiles]$ touch test
[sid@vmrhelu4 profiles]$ ls
ls: .: Permission denied
[sid@vmrhelu4 profiles]$ exit
exit
[root@vmrhelu4 sales]# chmod o=rx profiles/
[root@vmrhelu4 sales]# su sid
[sid@vmrhelu4 sales]$ cd profiles/
[sid@vmrhelu4 profiles]$ ls -l
total 4
-rw-r--r-- 1 sid accounts 0 Mar  8 08:11 test
[sid@vmrhelu4 profiles]$ █

[sid@vmrhelu4 profiles]$ █
-rw-r--r-- 1 sid accounts 0 Mar  8 08:11 test
total 4
[sid@vmrhelu4 profiles]$ ls -l
[sid@vmrhelu4 profiles]$ cd profiles\
[root@vmrhelu4 sales]# su sid
[root@vmrhelu4 sales]# chmod o=rx profiles\
```

Absolute Mode

- Permissions can also be set using numbers
- Using numbering scheme, the chmod command has three number places, that represent the three user types; user, group and others, respectively

example :*chmod 744*


When using chmod this way, all current permissions are replaced by the permissions you set

| Number | Meaning | Equivalent to |
|--------|---------|----------------------|
| 0 | --- | no access |
| 1 | --x | execute |
| 2 | -w- | write |
| 3 | -wx | write & execute |
| 4 | r-- | read |
| 5 | r-x | read & execute |
| 6 | rw- | read & write |
| 7 | rwX | read, write, execute |

umask - file mode creation mask

- What is umask command?
- - get or set the file mode creation mask
- If user want to check current umask value type the umask as a command on command prompt. It will display system current umask.
- Command –
 - umask
 - Example to set new umask value
umask 123

umask setting

 bgavkare@localhost:~

```
[bgavkare@localhost ~]$ umask
```

```
0002
```

```
[bgavkare@localhost ~]$ touch sample.txt
```

```
[bgavkare@localhost ~]$ ls -l sample.txt
```

```
-rw-rw-r-- 1 bgavkare bgavkare 0 Mar  6 11:19 sample.txt
```

```
[bgavkare@localhost ~]$ umask 123
```

```
[bgavkare@localhost ~]$ touch newfile.txt
```

```
[bgavkare@localhost ~]$ ls -l newfile.txt
```

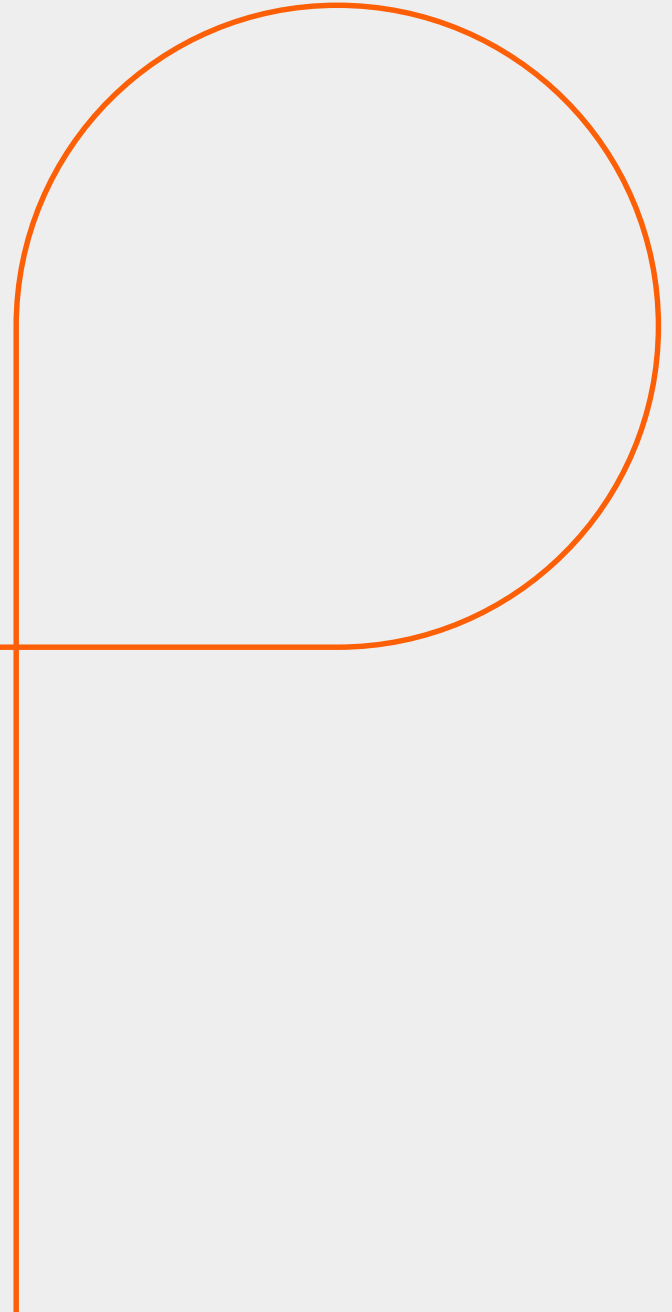
```
-rw-r--r-- 1 bgavkare bgavkare 0 Mar  6 11:20 newfile.txt
```

```
[bgavkare@localhost ~]$ █
```

Umask calculation

| Operations | Decimal Value | Binary Value |
|---|---------------|--|
| By Default file permission | - 6 6 6 | 1 1 0 1 1 0 1 1 0 |
| current system umask value | - 1 2 3 | 0 0 1 0 1 0 0 1 1 |
| Take the 1's complement of current system umask value | - | 1 1 0 1 0 1 1 0 0 |
| perform logical AND operation between 1's complement and default file permission values | && | <hr/> |
| Output of && operation is i.e. | | 1 1 0 1 0 0 1 0 0 r w - r - - r - - |

Links

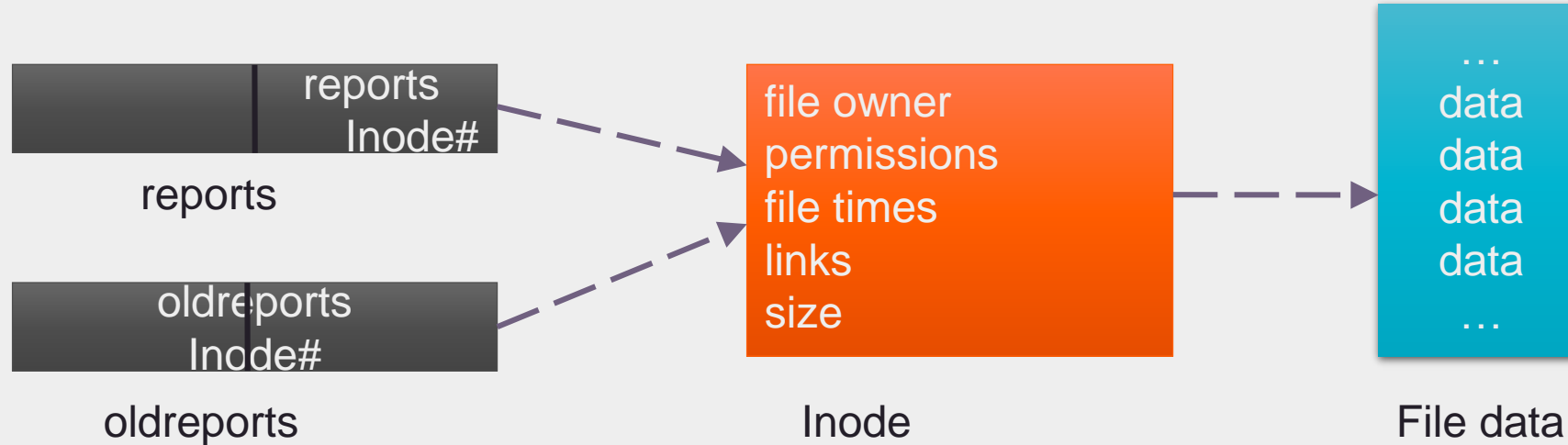


Links

- Links are like shortcuts in MS Windows and are extremely useful in creating convenient shortcuts to other files, or aliasing a command, etc
- There are two types of links in Unix
 - hard link
 - soft link or symbolic link

Hard Link

- Since all files are represented by an inode number, the files names internally point to the inode
- More than one filename can reference an inode number; these files are said to be hard linked together
- Any change made to the file through one link will reflect in the other, because filenames are just pointers to the same inode



Creating Hard Links

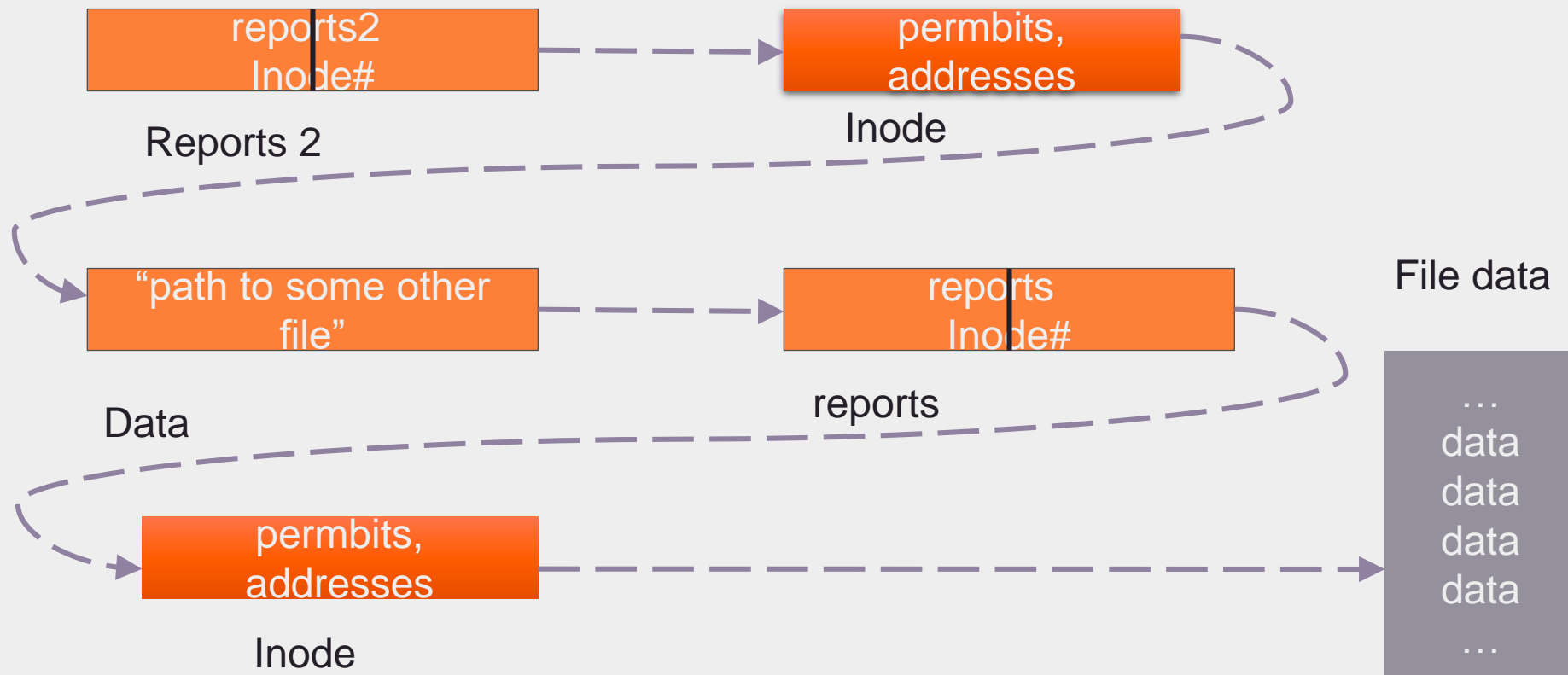
- Hardlinks are created through `ln` command as

In <sourcefile> <hardlink>

```
[root@vmrhelu4 links]# ls -li
total 8
261122 -rw-r--r-- 1 root root 12 Mar  8 02:51 sample.txt
[root@vmrhelu4 links]# ln sample.txt hardlink.txt
[root@vmrhelu4 links]# ls -li
total 16
261122 -rw-r--r-- 2 root root 12 Mar  8 02:51 hardlink.txt
261122 -rw-r--r-- 2 root root 12 Mar  8 02:51 sample.txt
[root@vmrhelu4 links]# ln hardlink.txt anotherhardlink.txt
[root@vmrhelu4 links]# ls -li
total 24
261122 -rw-r--r-- 3 root root 12 Mar  8 02:51 anotherhardlink.txt
261122 -rw-r--r-- 3 root root 12 Mar  8 02:51 hardlink.txt
261122 -rw-r--r-- 3 root root 12 Mar  8 02:51 sample.txt
[root@vmrhelu4 links]# rm sample.txt
rm: remove regular file `sample.txt'? y
[root@vmrhelu4 links]# ls -li
total 16
261122 -rw-r--r-- 2 root root 12 Mar  8 02:51 anotherhardlink.txt
261122 -rw-r--r-- 2 root root 12 Mar  8 02:51 hardlink.txt
[root@vmrhelu4 links]# cat anotherhardlink.txt
A text file
[root@vmrhelu4 links]#
```

Soft Link Or Symbolic Link

- A soft link is also called a symbolic link, and is a file that contains the name of another file
- Simply put, a symbolic link is like a pointer to the file's contents



Creating Symbolic Links

- Symbolic links can also be created through the *ln* command

ln -s <source file> <soft link name>

```
[root@vmrhelu4 links]# ls -li
total 8
261123 -rw-r--r--  1 root root 20 Mar  8 03:14 sample.txt
[root@vmrhelu4 links]# ln -s sample.txt softlink
[root@vmrhelu4 links]# ls -li
total 12
261123 -rw-r--r--  1 root root 20 Mar  8 03:14 sample.txt
261122 lrwxrwxrwx  1 root root 10 Mar  8 03:15 softlink -> sample.txt
[root@vmrhelu4 links]#
```

Hard Links

- Hard links point to the physical data
- All the hard links of a file will have the same inode number
- Any changes to the file will reflect in all the links
- The file data is not destroyed until the last link is deleted
- Hard links cannot be created across partitions
- You cannot hard link a directory since it might lead to recursion
- These are faster than symbolic links

Soft Links

- Soft links refer to the path of the file
- If the file is moved or deleted, the soft link becomes a dangling link
- Soft links of a file will have different inode numbers
- Deleting a soft link has no effect on the original file
- You cannot change the permissions of a soft link
- Soft links can be created across partitions
- Directories can have soft links
- These are slower than hard links

Links for objective multiple choice questions.

- <http://www.sanfoundry.com/linux-command-mcq-1/>
- <http://www.sanfoundry.com/linux-command-mcq-2/>
- <http://www.sanfoundry.com/linux-command-mcq-3/>
- <http://www.indiabix.com/computer-science/unix/>
- <http://www.avatto.com/computer-science/test/mcqs/questions-answers/unix/153/1.html>
- <http://www.gkseries.com/computer-engineering/unix/multiple-choice-questions-and-answers-on-unix-and-shell-programming>
- http://www.withoutbook.com/online_test.php?quiz=38&quesNo=10&subject=Top%2010%20UNIX%20Online%20Practice%20Test%20%7C%20Multiple%20Choice