Cat Command

Definition and purpose:

Cat command is used to display the content of a file or multiple files.

This command is more helpful in reading the log file or configuration files of the application or system.

Syntax:

```
Cat file-name1 file-name2
```

Example:

Cat /etc/passwd

The above command displays the user's details that are available on the machine.

The primary usage of Cat command is for displaying the file content on the terminal window but with Cat command, we can create files and add content also.

Most practical use is for:

- Checking the Application logs or system logs
- Concatenate two files or log files information and push to another file.
- Copying the content from one file to another file.
- Copying the content from one file to another file along with standard input from the terminal.
- To empty the file for having new content in it.

Below we have talked about few cat command options and their usage with practical examples.

Example-1:

Creating a new file with cat command:

Syntax:

```
“cat > filename”
```

Once the above command is typed and press enter. We see the cursor blinking on the next line, i.e, whatever we may type here, that will be redirected to that file.

Check out the below example,

```
[rreddy@abclearn abclearn_dir1]$ cat > abc_lab3.txt
Hi
this information saves in new file
3rd line
- must be working
this is 5th line for testing
```

Note:

- we have to press “ctrl+d” to come out of file writing.
- If the file is not created already then it will be created with the user credentials.
- If the file is already existing, then its content is overwritten.
- So using “>” symbol we can redirect the information to given file.

We can use the same cat command to display files content.

To display content of abc_lab3.txt file,

```
Cat abc_lab3.txt
```

We have created a new file. Now, what if we want to add more lines to that existing file.

Example-2:

Add new lines to existing file:

Syntax:

```
Cat >> filename
```

Note1:

- ">" symbol is used to redirect the information into a file.
- ">>" symbol will add new text at the end of existing file.If the file doesn't exist already, then it is created.

We can add more information to existing file at the end using ">>" symbol.

```
[rreddy@abclearnabclearn_dir1]$ cat >>abc_lab3.txt
this is going to be as 6th line
7th one
8th one
Let's see the file content and confirm, it is really added.
[rreddy@abclearnabclearn_dir1]$ cat abc_lab3.txt
Hi
this information saves in new file
3rd line
- must be working
this is 5th line for testing
this is going to be as 6th line
7th one
8th one
```

Example-3:

Copy a file's content to another file:

Syntax:

```
Cat source-file-name > target-file-name
```

As discussed earlier, ">" symbol will redirect the content which should be displayed on terminal window to another file.

So source file content is not removed at all. It is simply redirecting the file content to another file instead of on terminal.

We can use the cat command with ">" symbol to copy the content to another file and ">>" to append at the bottom of another file.

```
[rreddy@abclearnabclearn_dir1]$ cat abc_lab3.txt > abc_lab4.txt

[rreddy@abclearnabclearn_dir1]$ cat abc_lab4.txt
Hi
this information saves in new file
3rd line
- must be working
this is 5th line for testing this is going to be as
6th line
7th one
8th one
```

Example-4:

Display more than one file or multiple files output on terminal:

Cat command can concatenate one or more files content and show the output on the terminal.

Syntax:

```
Cat file-name1 file-name2
```

Let's see a practical example for better understanding.

```
[rreddy@abclearnabclearn_dir1]$ cat abc_lab1.txt abc_lab2.txt abc_lab3.txt
this line should replace the existing one
this is the first line in new file
Hi
this information saves in new file
3rd line
- must be working
this is 5th line for testing
this is going to be as 6th line
7th one
8th one
```

Example-5:

Redirect multiple files output to another file (or) concatenate more than one file output in another file:

Following the explanations are given in above examples, we will make use of ">" symbol for redirecting to file and ">>" symbol for adding the new content at the bottom of the target file.

Syntax:

```
Cat file-name1 file-name2 > file-name3

Cat file-name1 file-name2 >> file-name3
```

```
[rreddy@abclearnabclearn_dir1]$ cat abc_lab1.txt abc_lab2.txt abc_lab3.txt
> abc_lab4.txt

[rreddy@abclearnabclearn_dir1]$ cat abc_lab4.txt
this line should replace the existing one
this is first line in new file
Hi
this information saves in new file
3rd line
- must be working
this is 5th line for testing
this is going to be as 6th line
```

```
7th one
8th one
```

Example-6:

Display the line numbers of a file on terminal:

we have a couple of options with cat command to display output with line numbering.

Syntax:

```
Cat -n file-name
```

-n will display the line numbers including the empty lines.

Observe the below practical example and see line number 12 & 14. They are empty lines.

```
[rreddy@abclearnabclearn_dir1]$ cat -n abc_lab4.txt
1   this line should replace the existing one
2   this is first line in new file
3   Hi
4   this information saves in new file
5   3rd line
6   - must be working
7   this is 5th line for testing
8   this is going to be as 6th line
9   7th one
10  8th one
11  I am good
12
13  How are you
14
15  closing this file here
```

Example-7:

Display line numbers of a file but avoid empty lines:

For avoiding the empty lines from displaying in a file, we have command option is, “-b”

Syntax:

```
Cat -b file-name
```

Let’s look at the below cat command example, which completely avoids displaying the empty files.

```
[rreddy@abclearnabclearn_dir1]$ cat -b abc_lab4.txt
1  this line should replace the existing one
2  this is first line in new file
3  Hi
4  this information saves in new file
5  3rd line
6  - must be working
7  this is 5th line for testing
8  this is going to be as 6th line
9  7th one
10 8th one
11 I am good
12 How are you
13 closing this file here
```

Example-8:

Display or concatenate 2 files content with terminal one and standard input:

As we have discussed earlier, cat command comes with lot of options for variety of information.

We can take files information and concatenate it with terminal input.

Syntax :

```
Cat filename1 – filename2
```

“-“ option is to support input information from terminal or standard input also.

See the below example and try it out on terminal window,

```
[rreddy@abclearnabclearn_dir1]$ cat abc_lab1.txt - abc_lab2.txt
this line should replace the existing one--------------------------------
-->abc_lab1.txt file content
hi-----------------------------------------------------------------------
-->information typed on terminal
hi
new lines
new lines
sadfa
sadfa
this is first line in new file-------------------------------------------
--->abc_lab2.txt file content
```

Example-9:

Emptying the file having the content:

In such cases we need to empty the files to reduce the utilization of the storage in the machine.

Syntax:

```
Cat /dev/null > File-name
```

```
[rreddy@abclearnabclearn_dir1]$ cat /dev/null > abc_lab2.txt
[rreddy@abclearnabclearn_dir1]$ cat abc_lab2.txt
```

Exploring more cat command options:

For more on command options and their understanding, check out the manual pages or help content given with cat command.

Syntax:

```
man cat
cat --help
info cat
```

Topics Summary

cp command

Definition and purpose:

The primary usage of cp command is to “copy” files or directory from one location to another.

On Linux and UNIX flavor machines cp command is one of the most used commands in the day to day activities.

Syntax:

```
cp source-file-name target-file-name
```

Example:

copying the file1 information to file2 within the same directory.

cp file1 file2

More about cp command and its options, please checks its manuals and help content.

Syntax:

```
- man cp
- cp –help
- info cp
```

Practical usages:

- Using the backup of a file or directory during installation and upgradations.
- Making a copy of a files content to another file.
- Moving the log files from one location to another location.

Example-1:

Copy content of a file to another file in the same directory:

This is the most practical use case of cp command.

Sometimes, we might want to do some changes to any application configuration file, then it's always a best practice to take a copy of that file content. This will act as a backup file.

Syntax:

```
Cp Source-file-name target-file-name
```

In the below given example, we want to copy "abc_lab1.txt" file content to "abc_lab1_copy.txt" .

```
[rreddy@abclearnabclearn_dir1]$cp abc_lab1.txt abc_lab1_copy.txt
[rreddy@abclearnabclearn_dir1]$ cat abc_lab1_copy.txt
this line should replace the existing one
```

Note:

- Here, an abc_lab1_copy.txt file is created newly. If this file already exists on the directory, then it will be overwritten by default.

Example-2:

Copy the file content but don't override the existing one:

We can use cp command in combination with "-n" option.

This file will copy the source file to destination file without disturbing or overwriting the original content of destination file.

Therefore the final content of the destination file is source file content plus destination file content.

Syntax:

```
cp -n source-file target-file
```

Let's observe a practical example.

There is a file and it is having all ftp user details. We have a copy of ftp user details of another system as well. Now, my task is to combine all the users details in the single file.

In this scenario, we will use this option "-n".

Example Output:

```
[rreddy@abclearnabclearn_dir1]$cp -n abc_ftp1.txt abc_ftp2.txt
cp: overwrite 'abc_ftp2.txt'? y
[rreddy@abclearnabclearn_dir1]$cat abc_ftp2.txt
```

Example-3:

Copy the file from one directory to another directory:

Syntax:

```
cp source-file-name Target-directory
```

Let us take a practical case,

As part of version upgrade for an application software, “app1_config.txt” and other configuration files in installation directory are going to be updated to new format.

So we want to take a copy of these files, in another directory “/opt/bin/App1_Config_backup/” for any roll back activity.

```
[rreddy@abclearnApp1_Config]$ cpapp1_config.txt
/opt/bin/App1_Config_backup/
```

Changing to target directory,

```
[rreddy@abclearnApp1_Config]$ cd /opt/bin/App1_Config_backup/
[rreddy@abclearnApp1_Config]$ ls
app1_config.txt
```

Note-1:

Here, App1_Config_backupdirectory should be created already. If it doesn’t exist, then will throw an error.

```
[rreddy@abclearnApp1_Config]$ cp app1_config.txt
/home/rreddy/abclearn_dir2/
```

Note-2:

If we miss the last “/” shown in the red mark, instead of copying the content to dir2, there will be a new file created with the name as abclearn_dir2 under rreddy directory.

Also note that, to copy a file or create a new file user should be having proper credentials to that parent directory.

Example-4:

Copy entire directory to another directory:

Syntax:

```
cp -r Source-directory-name targer-directory-name
```

We can use the cp command with option “-r” recursive and copy entire directories and its sub-directories to another directory.

```
[rreddy@abclearn~]$cp -r abclearn_dir1/ /tmp/
[rreddy@abclearn~]$ cd /tmp
[rreddy@abclearntmp]$ ls
```

Recursive copy means, first sub directories and their files are copied and in the last parent directory is copied.

Example-5:

Forcing the copy operation:

While copying the file from one location to the another location it will ask for the confirmation by using the cp command with '-f' option won’t prompt the confirmation.

Syntax:

```
cp -f source_file Target _directory
```

```
[rreddy@abclearnabclearn_dir1]$cp  -f abc_lab1.txt  /tmp
[rreddy@abclearn~]$ cd /tmp
[rreddy@abclearntmp]$ ls
```

Example-6:

Prompting when overwriting a file:

Whenever we are copying the content of one file to the another file there is a chance of losing the important data in the destination file.

By using the cp command with "-i" option we get confirmation alert while overwriting the destination file with the source file.

Syntax:

```
cp -i source_file Destination_file
```

```
[rreddy@abclearnabclearn_dir1]$cp -i abc_lab1.txt abc_lab1_copy.txt
cp: overwrite `abc_lab1_copy.txt'? y
[rreddy@abclearnabclearn_dir1]$cat abc_lab1_copy.txt
```

Example -7:

Copying the file to the directory without changing the attributes of the file:

While copying the file from one location to the another location the attributes like timestamp of the file in the target location will change.

Without disturbing the attributes of the file we can copy the file from source to target using "-p" option.

Syntax:

```
cp  -p source_file Destination_directory
```

Follow the following scenario to understand more.

```
[ rreddy@abclearn ~]# cat abc_lab1.txt
```

```
[rreddy@abclearn~]# ls -l abc_lab1.txt
-rw-r--r--. 1 rreddy rreddy 0 Mar 21 19:32 abc_lab1.txt
```

```
[rreddy@abclearn~]# cp abc_lab1.txt /tmp
[rreddy@abclearn~]# cd /tmp
[rreddy@abclearn~]# ls -l abc_lab1.txt
-rw-r--r--. 1 rreddy rreddy 0 Jul 26 15:21 abc_lab1.txt
```

More to practice:

-u, --update copy only when the SOURCE file is newer

then the destination file or when the

destination file is missing

-s, --symbolic-link make symbolic links instead of copying

-H follow command-line symbolic links in SOURCE

-l, --link hard link files instead of copying

-L, --dereference always follow symbolic links in SOURCE

-a, --archive same as -dR --preserve=all

--attributes-only don't copy the file data, just the attributes

--backup[=CONTROL] make a backup of each existing destination file

-b like --backup but does not accept an argument

Topics Summary

mv command

Linux mv command practical examples

In Linux and UNIX flavors, mv command has multiple advantages.

1. It can be used for moving the files from one directory to another directory. Like cut & paste option, we usually have on word documents.

2. It can also be used for renaming a file in a given directory.

Note:

- There is no other command in Linux/Unix for renaming the files & directories.
- mv command can't rename a directory.

Syntax:

```
mv  source-file-name   target-file-name
```

```
mv  source-file-name   target-directory
```

Let us understand this command practical use cases with below examples.

Example-1:

Rename a file in a directory:

```
[rreddy@abclearn abclearn_dir2]$ ls
abc_lab1.txt
```

```
[rreddy@abclearn abclearn_dir2]$ mv abc_lab1.txt newfile.txt
[rreddy@abclearn abclearn_dir2]$ ls
newfile.txt
```

Move a file from one directory to another directory

This mv command will remove the file from present directory and will made available in new directory.

```
[rreddy@abclearn abclearn_dir1]$ mv abc_lab1_copy.txt
/home/rreddy/ebclearn_dir2/
```

If we check under abclearn_dir1 then file should be removed,

```
[rreddy@abclearn abclearn_dir1]$ ls
abc_lab1.txt abc_lab2.txt abc_lab3.txt  abc_lab4.txt
```

the same file is available under abclearn_dir2.

```
[rreddy@abclearn abclearn_dir1]$ ls /home/rreddy/abclearn_dir2/
abc_lab1_copy.txt newfile.txt
```

Observe:

If we know the directory path, then check directory content using ls command, without changing into it.

Move one directory as subdirectory to another

Couple of more options to try

Promoting options

-u, --update

Move only when the SOURCE file is newer than the destination file or when the destination file is missing

-b like --Backup but does not accept an argument

Topics Summary

1.What is the use cases of mv command?can we rename a particular directory using mv command?

2.How to rename a file in a directory?

3.How to move a file from one directory to another directory?

4.How to make a directory as sub-directory to another? For ex: dir1 should be sub-directory for dir2.

WC Command

WC Command Examples

Practical purpose:

Most of the times, when we work with scripting programs,

we might encounter with a requirement of counting the number of lines of a file for writing the logic.

We may want to count the lines/words/characters of a command output.

This kind of requirements are best addressed by “wc command”.

wc command syntax:

```
wc file-name
wc file-name1 file-name2
```

Example-1:

Checking the lines,words and characters in a file:

```
[rreddy@abclearn abclearn_dir1]$ wc  abclearn_lab1.txt
1 7 42 abclearn_lab1.txt
```

From the output,

1st column represents number of **line** in file

2nd column represents number of **words** in file

3rd column represents number of **characters** in file.

Example-2:

Counting the words in a file or in a content:

```
[rreddy@abclearn abclearn_dir1]$ wc -w abclearn_lab1.txt
7 abclearn_lab1.txt
```

Example-3:

Counting the lines in a file or in an output:

```
[rreddy@abclearn abclearn_dir1]$ wc -l abclearn_lab4.txt
15 abclearn_lab4.txt
```

We can also count number of line in more than one file, also.

```
[rreddy@abclearn abclearn_dir1]$ wc -l abclearn_lab4.txt abclearn_lab3.txt
15 abclearn_lab4.txt
8 abclearn_lab3.txt
23 total
```

Example-4:

Printing the byte counts of the file:

By using "-c" option with wc command we can print the byte counts of a particular file.

```
[rreddy@abclearn abclearn_dir1]$ wc -c abclearn_lab4.txt
1816 abclearn_lab4.txt
```

Example-5:

Printing the length of largest line:

By using "-L" option with wc command we can print the length of largest line of a particular file.

```
[rreddy@abclearn abclearn_dir1]# wc -L  abclearn_lab4.txt
79 abclearn_lab4.txt
```

Example -6:

Printing the new line count or Line count:

By using "-l" option with wc command we can print the new line (/n or enter) count of the particular file.

```
[rreddy@abclearn abclearn_dir1]$ wc -l abclearn_lab4.txt
39 abclearn_lab4.txt
```

Topics Summary

1.What are some real time situations in which we use wc command?

2.How to check lines, words and characters of a file?

3.How to count the words in a file or an input content?

4.Which command we use for counting number of lines in a file?

5.How to print the byte or character count of a file?

6.What is the command to print length of longest line in a file?

rmdir and rm command

Linux rmdir and rm command examples

In this section, we will look at more on how to remove a file or directory on the Linux/UNIX flavor machines.

Let's start with rmdir command.

This is used for removing of an empty directory. An empty directory means, having no files and subdirectories in it.

Syntax:

```
rmdir directory-name
```

rmdir command execution,

```
[rreddy@abclearn abclearn_dir1]$rmdirabclearn_dir2/
```

In practical, we don't use this rmdir command so frequently. Main reason is,
Most of the times, every directory will be having at least few files and subdirectories under it. So to use this command, we have to manually eliminate each and every file and subdirectory of it. which is a tedious task.

The best solution is using “rm command” with “-rf” option.

rm command:

rm command is primarily used to remove single or multiple files under a directory.

Syntax:

```
rm file-name
```

Let’s understand some of its practical usages with examples.

Example-1:

Removing a particular file:

Removing a file is a straightforward requirement and there are so many of the cases for this.

Some practical usage cases,

We have worked on a sample file and want to delete after its usage.

We might find a corrupted configuration file and want to remove it and copy a new file from backup.

Syntax:

```
rm filename
```

sample rm command output is given below,

```
[rreddy@ebclearn ebclearn_dir1]$rm abc_lab1.txt
```

The one problem is, it will be removing the file without even prompting and once the file is deleted in Linux/UNIX flavor OS, we can’t pull it back.

So always best practice is to prompt for deletion of a file.

rm command with “-i” option, will prompt the user whether to delete a file or not.

```
[rreddy@abclearn abclearn_dir1]$rm -i abc_lab2.txt
rm: remove regular file ‘abc_lab2.txt’? y
```

Example-2:

Force removing a file:

We can remove the file using "-f" command with rm command it won't prompt for confirmation to delete it or not.

Syntax:

```
rm -f filename
```

Observe the following case for the proper understanding.

```
[rreddy@abclearn abclearn_dir1]$rm -f abc_lab2.txt
```

Example-3:

Removing multiple files at a time:

For removing multiple files, we will use rm command followed by file names

Syntax:

```
rm -i file-name1 file-name2
```

Observe the following usage for more understanding.

```
[rreddy@abclearn abclearn_dir1]$rm -i abc_lab2.txt abc_lab3.txt
```

For example:

If we want to delete all files under a particular directory using rm command, we can use directory name followed by a star as indicated below.

```
$ rm -rf Directory_Name/*
```

Practical usage case,

We might want to remove files from /tmp directory for memory free.

```
[rreddy@abclearn~]$ rm -rf /tmp/*
```

Example-4:

Removing a directory with files and subdirectories under it:

We can remove the directory including directories and files in the particular directory using rm command with "-r" option in this r indicates recursive.

Syntax:

```
rm -rf directory-name
```

Practical usage case,

Suppose a package is uninstalled/ removed from the system but the files and directories used by that package will be there and consume some amount of storage. to free that storage we will remove that directory with "rm -rf' command

```
[rreddy@abclearn~]$ rm -rf  abc_lab
```

Example-5:

Removing the files having the same extension:

In some cases, we need to remove all files having the same extension. we use wild cards in these scenarios.

to understand more follow the following scenario.

```
[rreddy@abclearn~]$ ls *.txt
a.txt  b.txt  c.txt  d.txt
```

```
[rreddy@abclearn~]$ rm -i *.txt
rm: remove regular empty file `a.txt'? y
rm: remove regular empty file `b.txt'?
```

Possible issues we might face while removing a file or directory:

1. We need to have proper privileges on that directory or file in order to delete the file or directory.

2. Sometimes the filename which we are going to delete may end with space or tab. Include those white space characters while deleting the file.

Topics Summary

su command

su command practical examples

As we work through Linux administration activities, we most of the times have to switch between one user credentials to another user credentials.

In this section, we will look at those requirement and command operations.

su command:

su is abbreviated as “switch user”.

Syntax:

```
su user-account-name
```

Throughout this section, we will understand about this command implementation with different example scenarios.

Note:

sudo & su both commands are different in nature.

To understand more about sudo command, **How can use sudo command for file executions?**

Before we get started with examples, let us understand specific lookout areas of user identification.

Once we login to terminals, let’s check with what credentials I have logged into and what is my default directory.

```
[rreddy@abclearn rreddy]$pwd
/home/rreddy
```

Looking at the command terminal in XXX shell,

- We can see userid specified before the server name.

Also “$” dollar symbol indicates that we have logged in as a non-root user.

Example-1:

Switch from one user to another user:

Using su command, we can switch from one user to another user account.

From rreddy, let’s switch to another user account called user1. Type the password.

```
[rreddy@abclearn~]$su user1
Password:
```

```
[user1@abclearn rreddy]$ whoami
user1
```

we can see that from rreddy, now I have logged in as user1.

Checking with pwd command,

```
[user1@abclearn rreddy]$pwd
/home/rreddy
```

Note:

In the above output, only user account got switched but not logged into new user’s home directory.

Example-2:

Switch user with its login user’s home directory:

With the same su command, if we specify “-“ then we will be in new user home directory by default.

Syntax:

```
su – user-account-name
```

From rreddy, let’s switch to another user account called user1.

```
[rreddy@abclearn~]$su - user1
Password:
Last login: Fri July 15 20:38:37 EST 2016 on pts/1
```

```
[user1@abclearn user1]$whoami
user1
```

```
[user1@abclearn rreddy]$pwd
/home/user1
```

Note:

landing on to user's home directory may have advantages like
we can directly start accessing the files and directories under his home directory. This will avoid another "cd" command typing.

Also, assigns the default shell for this user.

```
[user1@abclearn ~]$ echo $SHELL
/bin/bash
```

How to check landing user's home directory and default shell information?

From /etc/passwd configuration file, we can understand each user default home directory and default shell.

```
[rreddy@abclearn~]$ grep user1 /etc/passwd
user1:x:1001:1001::/home/user1:/bin/bash
```

To come back to previous user account:

We can type in exit command to come out of user session. It will take you to the previous logged in user session,

Or else, we can again use su command to switch to the corresponding new user session.

```
[user1@abclearn rreddy]$ exit
exit
```

Note:

o For a root user, the password is not required while switching to another user. By default, he/she can switch to any user they want.

o If we have switched users a multiple number of times, then for each exit type control goes back to the previous user account.

Example-3:

Changing the default user login shell:

Syntax:

su --shell shell-path user-account-details

Practical usage:

In Shell scripting programing….

Another example….

Sample command output,

```
[rreddy@abclearn~]$su --shell /bin/sh user1
Password:
sh-4.2$ echo $SHELL
/bin/sh
```

Example-4:

Switching to root user account:

Just like switching to a normal user, we can specify “root” user name for switching to it.

But, there is a special case also,

even if we don't specify any user account details after su command, then by default, it will expect root account credentials.

Syntax:

```
Su – root
su –
```

```
[rreddy@abclearn~]$su -
Password:
Last login: Fri Jan 15 02:31:24 EST 2016 on pts/3
[root@abclearn~]#pwd
/root
```

Note:

A root user can switch to any other account without even knowing the password of that user.

```
[root@abclearn~]#su user1
[user1@abclearn ~]$whoami
User1
```

Topics Summary

1.How to switch from one user to another user in Linux?

2.How to switch back to previous user account from present one? Or come out of present use session?

3.How to switch to new user and landing into user's home directory?

4.How can we change the new user SHELL form the default when logging in?

5.How can we switch to root user account in Linux environment?

Ls command

Ls command is used to list the files and subdirectories under a specific directory. It is one among the regularly used commands while doing administration of a server.

Syntax:

```
ls
ls -l
ls -ltr
```

Above given are a couple of options which are most regularly used with ls command.

We will see some ls practical examples below for better understanding of it.

Example-1:

Listing the files and directories:

By typing normal ls command on the terminal window, will list all of the files and subdirectories under a directory.

Sample Usage case,

Assume that we have a software package unzipped on Linux/Unix flavor machine. Now, to see what are the files and sub-directories are available within that we can use this command.

We can take a numerous number of examples for "ls command", above is just a sample case. Normal "ls command" output is,

```
[root@sys1 repo]# ls
EFI         media.repo                      RELEASE-NOTES-or-IN.html
EULA        Packages                        RELEASE-NOTES-pa-IN.html
EULA_de     README                          RELEASE-NOTES-pt-BR.html
EULA_en     RELEASE-NOTES-as-IN.html        RELEASE-NOTES-ru-RU.html
EULA_es     RELEASE-NOTES-bn-IN.html        RELEASE-NOTES-si-LK.html
```

Example-2:

Long listing of files and directories:

From the normal “ls command”, we could see only list of things available but it won’t show up more details about it. Such as,

Whether it is a file or directory?

Who are the owner & group ownership of that file or directory?

What are the files permissions level and last modified dates? etc

We can see all of the above information from using “ls -l” command option.

Syntax:

```
ls -l
```

A sample command output is given below,

```
[root@sys1 repo]# ls -l
total 3120
-r--r--r--. 1    root   root    11414   Jan  31   2013   TRANS.TBL
dr-xr-xr-x. 3    root   root    4096    Jan  31   2013    Server
dr-xr-xr-x. 3    root   root    4096    Jan  31   2013
 ScalableFileSystem
-r--r--r--. 1    root   root    3211    Jan  29   2013    RPM-GPG-KEY-
redhat-release
-r--r--r--. 1    root   root    3375    Jan  29   2013    RPM-GPG-KEY-
redhat-beta
dr-xr-xr-x. 3    root   root    4096    Jan  31   2013    ResilientStorage
drwxr-xr-x. 2    root   root    4096    Feb   8   13:50   repodata
```

From “ls -l” command output we can make out,

Starting from left,

The first character explains whether it is a file or directory.

Next, 9 characters explain what permission levels are for the user, primary group and others in sequence.

Next character talks about file inode information.

Who are the primary owner and group owner of the files is explained?

File size and next time/date of file modified information.

Also, try,

```
ls -m
ls -x
```

Example-3:

Files with reverse order:

To display the files in reverse chronological order. We can use "-r" option with ls command.

Syntax:

```
Is-Ir
```

Note:

"-r" option should always use with "-l" option for better results.

One can observe that based on the file alphabetical order we can see the files organized. Sample output is given below.

```
[root@sys1 repo]# ls -l
total 3120
-r--r--r--. 1    root    root      11414     Jan 31    2013   TRANS.TBL
dr-xr-xr-x. 3    root    root       4096     Jan 31    2013   Server
dr-xr-xr-x. 3    root    root       4096     Jan 31    2013
ScalableFileSystem
-r--r--r--. 1    root    root       3211     Jan 29    2013   RPM-GPG-KEY-
redhat-release
```

```
-r--r--r--. 1    root    root       3375     Jan 29    2013   RPM-GPG-KEY-
redhat-beta
dr-xr-xr-x. 3    root    root       4096     Jan 31    2013
ResilientStorage
drwxr-xr-x. 2    root    root       4096     Feb 8     13:50  repodata
```

Example-4:

Files and directories with timestamp and reverse timestamp:

For displaying the files and directories based on timestamp, we can use "-t" option with ls command.

Syntax:

```
Ls -lt
```

Sample command output is given below,

```
[root@sys1 repo]# ls -lt
total 3120
drwxr-xr-x. 2        root   root      4096   Feb 8   13:50  repodata
drwx------. 2        root   root     16384   Feb 2   12:01  lost+found
-r--r--r--. 1        root   root     11414   Jan 31  2013   TRANS.TBL
dr-xr-xr-x. 3        root   root      4096   Jan 31  2013   images
dr-xr-xr-x. 2        root   root    253952   Jan 31  2013   Packages
dr-xr-xr-x. 3        root   root      4096   Jan 31  2013
HighAvailability
dr-xr-xr-x. 3        root   root      4096   Jan 31  2013   LoadBalancer
```

We can combine -r option with -t timestamp, for files and directories in reverse timestamp.

Syntax:

```
ls -ltr
ls -l -t -r
```

This is most used on day to day life, because it gives the recent modified files and directories information in the top order. Keep note of this for sure.

Sample ls -ltr command output is,

```
[root@sys1 repo]# ls -ltr
total 3120
drwxr-xr-x. 2    root   root     4096     Feb 8     13:50  repodata
drwx------. 2    root   root    16384     Feb 2     12:01  lost+found
-r--r--r--. 1    root   root    11414     Jan 31    2013   TRANS.TBL
dr-xr-xr-x. 3    root   root     4096     Jan 31    2013   images
dr-xr-xr-x. 2    root   root   253952     Jan 31    2013   Packages
dr-xr-xr-x. 3    root   root     4096     Jan 31    2013
HighAvailability
dr-xr-xr-x. 3    root   root     4096     Jan 31    2013   LoadBalancer
```

Example-5:

Files and directories with human readable size format:

As we have discussed, ls -l command will display the files and directories size information also but they are in blocks format. We need to convert them into kilo bytes and Mega bytes etc to understand clearly.

Instead, we can use "-h" option with ls command to display the sizes in human readable format.

Syntax:

```
ls -lh
```

sample command output is given below,

```
[root@sys1 repo]# ls -lh
total 3.1M
dr-xr-xr-x. 3          root    root    4.0K    Jan 31     2013    EFI
lrwxrwxrwx. 1          root    root       7    Jan 31     2013    EULA ->
EULA_en
-r--r--r--. 1          root    root      11K   Nov 7      2012    EULA_de
-r--r--r--. 1          root    root     8.6K   Nov 7      2012    EULA_en
-r--r--r--. 1          root    root      11K   Nov 7      2012    EULA_es
```

```
-r--r--r--. 1          root    root      11K   Nov 7      2012    EULA_fr
-r--r--r--. 1          root    root      11K   Nov 7      2012    EULA_it
-r--r--r--. 1          root    root      13K   Nov 7      2012    EULA_ja
-r--r--r--. 1          root    root     9.7K   Nov 7      2012    EULA_ko
-r--r--r--. 1          root    root     9.8K   Nov 7      2012    EULA_pt
-r--r--r--. 1          root    root     7.2K   Nov 7      2012    EULA_zh
```

Let's remove the -l option and observe the output,

Ls -h command output here….

Similar to above we can try,

ls -s To display file sizes in bytes

ls -k To display file sizes in kilo bytes

ls -m To display file sizes in mega bytes

By using ls –s command we can do this.

```
[root@sys1 repo]# ls -s
total 3120
4    EFI 92                RELEASE-NOTES-fr-FR.html
0    EULA 80               RELEASE-NOTES-gu-IN.html
12   EULA_de 136           RELEASE-NOTES-hi-IN.html
12   EULA_en 84            RELEASE-NOTES-it-IT.html
12   EULA_es 100           RELEASE-NOTES-ja-JP.html
12   EULA_fr 156           RELEASE-NOTES-kn-IN.html
12   EULA_it 88            RELEASE-NOTES-ko-KR.html
16   EULA_ja 164           RELEASE-NOTES-ml-IN.html
12   EULA_ko 140           RELEASE-NOTES-mr-IN.html
```

Example -6:

Listing the hidden files and directories:

Regular ls command will not display any hidden files and sub directories under a directory. Let's first understand,

what is a hidden file or directory? And Why we need that in first place?

In a day to day activities we might want to hide critical/important files and directories from regular user. This is just to avoid any accidental changes to those files etc.

for example, some application specific configuration files.

To hide a file or directory, we need to use "." Before file/directory name.

Example, .profile, .bash_profile etc

To view these hidden files and directories, we can use "-a" option with ls command.

Syntax:

```
ls -a
ls -la
ls -lat
```

Sample command output is given below,

```
[root@sys1 repo]# ls -a
. lost+found  RELEASE-NOTES-or-IN.html
.. media.repo RELEASE-NOTES-pa-IN.html
EFI Packages  RELEASE-NOTES-pt-BR.html
EULA README   RELEASE-NOTES-ru-RU.html
```

Example-7:

Listing out inode numbers with files and directories:

If you are new to inode numbers, then visit this, what are inode numbers? How they are associated with files and directories structure?

Syntax:

```
ls -i ls -li
```

A usage case,

When do we check inode numbers for a files?

Practical command output is given below,

By using ls –I command we can do this

```
 [root@sys1 repo]# ls -i
128001     EFI 3780           RELEASE-NOTES-fr-FR.html
12     EULA 3781              RELEASE-NOTES-gu-IN.html
13     EULA_de 3782           RELEASE-NOTES-hi-IN.html
14     EULA_en 3783           RELEASE-NOTES-it-IT.html
15     EULA_es 3784           RELEASE-NOTES-ja-JP.html
16     EULA_fr 3785           RELEASE-NOTES-kn-IN.html
17     EULA_it 3786           RELEASE-NOTES-ko-KR.html
18     EULA_ja 3787           RELEASE-NOTES-ml-IN.html
19     EULA_ko 3788           RELEASE-NOTES-mr-IN.html
20     EULA_pt 3789           RELEASE-NOTES-or-IN.html
21     EULA_zh 3790           RELEASE-NOTES-pa-IN.html
```

Topics Summary