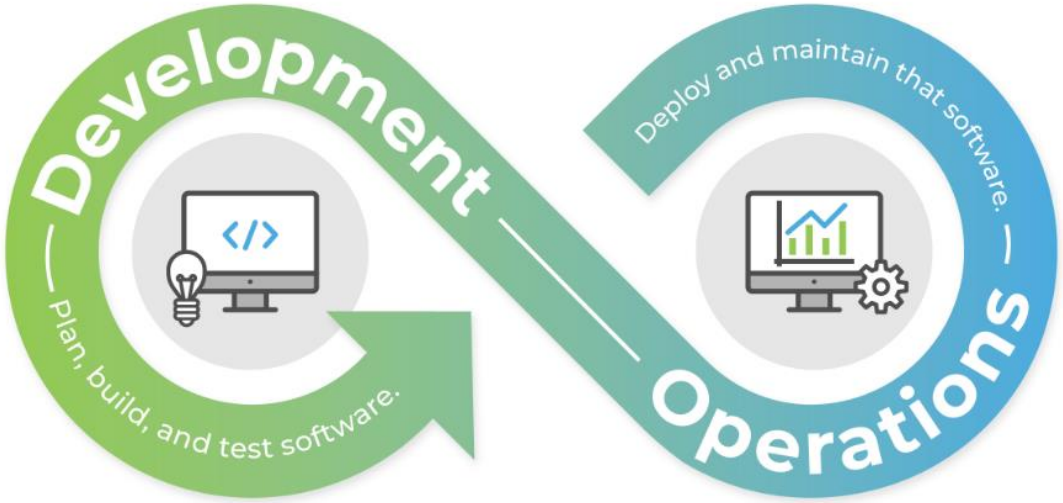
DevOps
01 Linux Basics

Linux Commands & More - Part 1



Agenda for this session

- Linux Commands – Basic Commands
- Working with Files and Folders, Changing Permissions & Ownership
- Types of Links – Soft Links and Hard Links

Linux Commands

Linux Command - Structure

$ command --option argument

Command: Command/program that does one thing

Options: Change the way a command does that one thing

- Short form: Single-dash and one letter (Example: ls –a)
- Long form: Double-dash and a word (Example: ls --all)

Argument: Provides the input/output that the command interacts with.

Example: $cal –j 3 2000

Commands – Small programs that do one thing well

“Many UNIX programs do quite trivial things in isolation, but, combined with other programs, become general and useful tools.”

(From the book ‘The Unix Programming Environment’, Kernighan and Pike)

Simple commands

```
raja@LAPTOP-FV4AF8PS:~$ whoami
raja
raja@LAPTOP-FV4AF8PS:~$ hostname
LAPTOP-FV4AF8PS
raja@LAPTOP-FV4AF8PS:~$ echo $HOME
/home/raja
raja@LAPTOP-FV4AF8PS:~$ echo my login is $(whoami)
my login is raja
raja@LAPTOP-FV4AF8PS:~$ date
Sat Aug 12 10:02:15 IST 2023
raja@LAPTOP-FV4AF8PS:~$ cal
Command 'cal' not found, but can be installed with:
sudo apt install ncal
raja@LAPTOP-FV4AF8PS:~$ dir
doc
raja@LAPTOP-FV4AF8PS:~$ close
Command 'close' not found, did you mean:
  command 'gclose' from deb gnustep-gui-runtime (0.29.0-2build1)
Try: sudo apt install <deb name>
raja@LAPTOP-FV4AF8PS:~$
```

printenv displays environment variables

Environment Variables - Examples

SHELL=/bin/bash

NAME=LAPTOP-FV4AF8PS

PWD=/home/raja

LOGNAME=raja

HOME=/home/raja

LANG=C.UTF-8

alias

```
raja@LAPTOP-FV4AF8PS:~$ cls
Command 'cls' not found, but there are 18 similar ones.
raja@LAPTOP-FV4AF8PS:~$ alias cls='clear'
raja@LAPTOP-FV4AF8PS:~$
```

alias command is used to provide an alias name to an existing command.

`clear`	clears the screen.
`alias cls=‘clear’`	provides an alias ‘cls’ to clear command.

Now, try

`cls`

Working with files and folders (directories) - Simple commands

pwd - displays the present working directory (also known as ‘print current directory’

ls - list files (similar to ‘DIR’ command in Windows but has several options)

cd - change directory (to move from one directory to another)

Special characters interpreted by the shell

~ - home directory

. current directory

.. parent directory

* wildcard matching any file name

? wildcard matching any character

Changing directory and listing files/directories

cd /usr/local/lib	Change directory to /usr/local/lib
cd ~	Change to home directory (could just type ‘cd’)
pwd	Print working (current) directory
cd ..	Change directory to the “parent” directory
cd /	Change directory to the “root”
cd ../my	Move one level above (to parent directory) and then move to ‘my’ directory
ls –d uni*	List only the directories starting with “uni”

ls command – more examples

ls -a	List all files, including hidden files
ls -ld *	List details about a directory and not its contents
ls -F	Put an indicator character at the end of each name
ls –l	Simple long listing
ls –la	long listing of files including hidden files
ls –lR	Recursive long listing
ls –lS	Sort files by file size
ls –lt	Sort files by modification time (very useful!)

Additional Linux commands

cp [file1] [file2]	copy file
mkdir [name]	make directory
rmdir [name]	remove (empty) directory
mv [file] [destination]	move/rename file
rm [file]	remove (-r for recursive)
file [file]	identify file type
less [file]	page through file

Try these commands too…

head -n N [file]	display first N lines
tail -n N [file]	display last N lines
ln –s [file] [new]	create symbolic link
cat [file] [file2…]	display file(s)
tac [file] [file2…]	display file in reverse order
touch [file]	update modification time
od [file]	display file contents, esp. binary

Additional examples

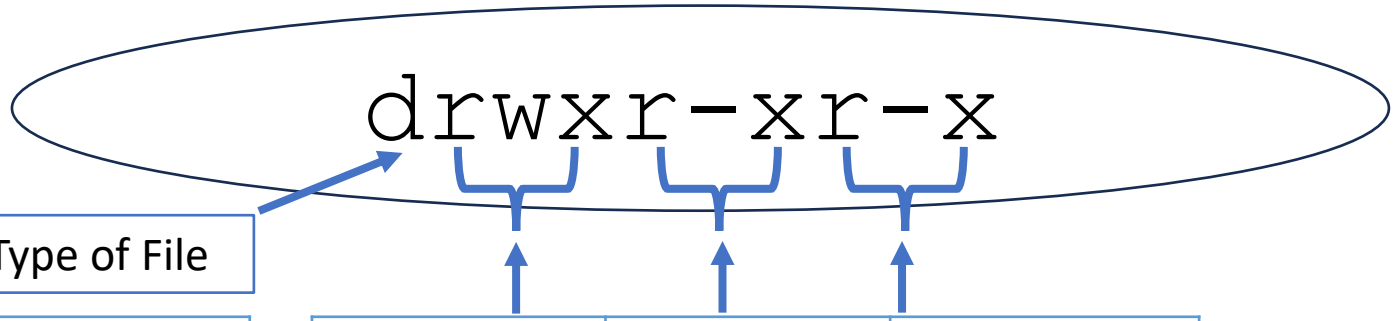
```
cd #                                The same as cd ~
mkdir test
cd test
echo ‘Hello everyone’ > myfile.txt
echo ‘Goodbye all’ >> myfile.txt
less myfile.txt
mkdir subdir1/subdir2                          # Fails. Why?
mkdir -p subdir1/subdir2                                  # Succeeds
mv myfile.txt subdir1/subdir2
cd ..
rmdir test                              # Fails. Why?
rm –rf test # Succeeds
```

The ls command

ls -l gives the long listing of files

```
drwxr-xr-x 6   ananth   trguser     1024   Aug 8 13.30 personal
-rw------- 1   ananth   trguser     1024   Aug 8 13.30 contacts
```

First 10 characters (in the output of ls -l)



Type of File

-	normal file
d	directory
l	symbolic link
p	named pipe
b	block device
c	character device
s	socket

owner	group	others
Permissions of the file owner	Permissions of the group	Permissions for everyone else

The 2nd column

ls -l gives the long listing of files

```
drwxr-xr-x 6   ananth   trguser     1024   Aug 8 13.30 personal
-rw-------1    ananth   trguser     1024   Aug 8 13.30 contacts
```

The second column is the number of links to the file *i.e.,* (more or less) the number of names there are for the file. Generally an ordinary file will only have one link, but a directory will have more, because you can refer to it as “personal“, “personal/.'' where the dot means ``current directory'', and if it has a subdirectory named ``subdir'', ``personal/subdir/..'' (the ``..'' means ``parent directory'').

The 3rd and 4th columns

ls -l gives the long listing of files

```
drwxr-xr-x 6   ananth   trguser     1024   Aug 8 13.30 personal
-rw------- 1   ananth   trguser     1024   Aug 8 13.30 contacts
```

Owner Group

The third and fourth columns are the user who owns the file and the Unix group of users to which the file belongs.

All columns

ls -l gives the long listing of files

```
drwxr-xr-x 6   ananth   trguser   1024   Aug 8 13.30 personal
-rw------- 1   ananth   trguser   1024   Aug 8 13.30 contacts
```

Type of files

Permissions (Owner, Group, Others)

Owner

Group

Size in bytes

Modification date & time

Name of the file or directory

history command

`history` displays command history

Using the up ↑ and down ↓ arrows choose a previously used command

To redo your last command, try !!

To go further back in the command history try !, then the number as shown by history (e.g., !5). Or, !ls, for example, to match the most recent 'ls' command.

Try left ← and right → arrows on the command line

Help on commands

Try these commands

date --help

date --help |more

man date

info date

help

man bash

Try ‘man’ with ‘less’

```
man less
```

This helps you scroll through the text displayed on the screen by using shortcut keys. For example,

Space or f for page forward, b for page backward, < to go to first line of file, > to go to last line of file, / to search forward (n to repeat), ? to search backward (N to repeat), h to display help, q to quit help

Changing Directory / File Permissions & Ownerships

Three types of file permissions

1. Read - allows a user to open and read the content of the file
2. Write - allows a user to edit content or rename or remove the file
3. Execute - determines if the user can execute the file

Three types of directory permissions

1. Read - allows a user to view the directory's contents
2. Write - allows a user to create new files, rename files or delete files in the directory
3. Execute - determines if the user can enter (cd) into the directory or run a program or script.

Summary – File and Directory Permissions

Permissions	File	Directory
Read	Read file contents	Read directory contents
Write	Change file contents, rename file, remove file	Create new files, rename or delete files in the directory
Execute	Execute the file	Enter the directory and/or run a program or script in that directory

File access levels

1. Owner Permissions
2. Group Permissions
3. Others Permission (everyone other than the owner and group of the owner)

Use the following command to know file or directory permissions.

```
ls -l
```

ls -l gives the long listing of files

```
drwxr-xr-x 6   ananth   trguser   1024   Aug 8 13.30 personal
-rw------- 1   ananth   trguser   1024   Aug 8 13.30 contacts
```

Type of files

Permissions (Owner, Group, Others)

Owner

Group

Size in bytes

Modification date & time

Name of the file or directory

Permission - representation

Binary	Octal	Permission
000	0	- - -
001	1	- - x
010	2	- w -
011	3	- w x
100	4	r - -
101	5	r - x
110	6	r w -
111	7	r w x

777 means rwxrwxrwx

654 means rw-r-xr--

744 means rwxr--r--

Changing permissions (chmod command)

$ chmod 644 contacts

When you execute this command, the permissions of contacts changes to rw- r-- r--

chmod command – other options

chmod g–w,o–r file1.txt

This will remove ‘write’ permission from group, and ‘read’ permission from ‘others’ on file.txt.

chmod u + x,g + x file1.txt

This will add ‘execute’ permission to user(owner), and add ‘execute’ permission to group on file.txt

Note: These is no blank space in the second argument of this command. After chmod there is a blank space. After that, it is a single string. There is a blank space before preceding file1.txt.

chmod command - other options

chmod u=rwx,g=r,o= file1.txt

This sets rwxr----- on file1.txt (user or owner can rwx. Group can read, others do not have any permission

Note: These is no blank space in the second argument of this command. After chmod there is a blank space. After that, it is a single string. There is a blank space before preceding file1.txt.

chmod command – other options

These are valid too.

chmod u = rwx,g–w,o–r file1.txt

chmod u = rwx,g–w + x,o–r file1.txt

Changing file ownership

chown command is used to change file ownership

chown userB file1.txt

This command changes the owner of file1.txt to userB

Types of Links (Soft links, &Hard links)

inode

Linux users create directories and files. Linux stores information about a file in ‘inode’.

‘inode’ is a structured data (or data structure) that keeps track of all information about a directory or file.

When a user wants to access a file called ‘samplefile’, Linux searches a table called ‘inode’ table to find the ‘inode’ of ‘samplefile’.

The ‘inode’ information of ‘samplefile’ provides the exact location of the file on the hard disk, in addition to other details such as owner, size, etc.,

Read: https://www.slashroot.in/inode-and-its-structure-linux

inode of a file

Mode	
File Owner	
Size in Bytes	
Time Stamps (creation date & time, modification date & time	
Direct Blocks	Points to first 12 data blocks. Up to 48K file size. (12 x 4K = 48 K)
Indirect Blocks	Points to 1024 data blocks. Up to 4MB file size. (1024 x 48K = 4 MB)
Double Indirect Blocks	Points to 1024 data blocks. Up to 4GB file size. (1024 x 4M = 4 GB)
Triple Indirect Blocks	Points to 1024 data blocks. Up to 4TB file size. (1024 x 4 GB = 4 TB)

This is based on the default block size of 4K.

inode structure of a directory

Inode No 3470036	
. (DOT)	3470036
..(DOT DOT)	3470017
Folder 1	3470031
File 1	3470043
File 2	3470023
Folder 2	3470024
File 3	3470065

Sub-directories and files

inode numbers

Use ‘ls -ia’ command to know inode numbers

ls –ia

```
3633723 .                3633786 faillog     3634889 rpmpkgs.3
3633697 ..               3634727 gdm         3634893 rpmpkgs.4
3634833 acpid            3633883 httpd       3633813 samba
```

Hard Links

- Every file on the Linux filesystem starts with a single hard link. The *link* is between the filename and the actual data stored on the file system.
- Use ln command to create an hard link to a file

```
ln <original file path> <new file path>
ln <original file name> <new file name>

ln /home/user1/fileA  /home/user1/doc/fileB
ln fileA fileB
```

Read: https://www.redhat.com/sysadmin/linking-linux-explained
https://www.redhat.com/sysadmin/inodes-linux-filesystem

Hard Links

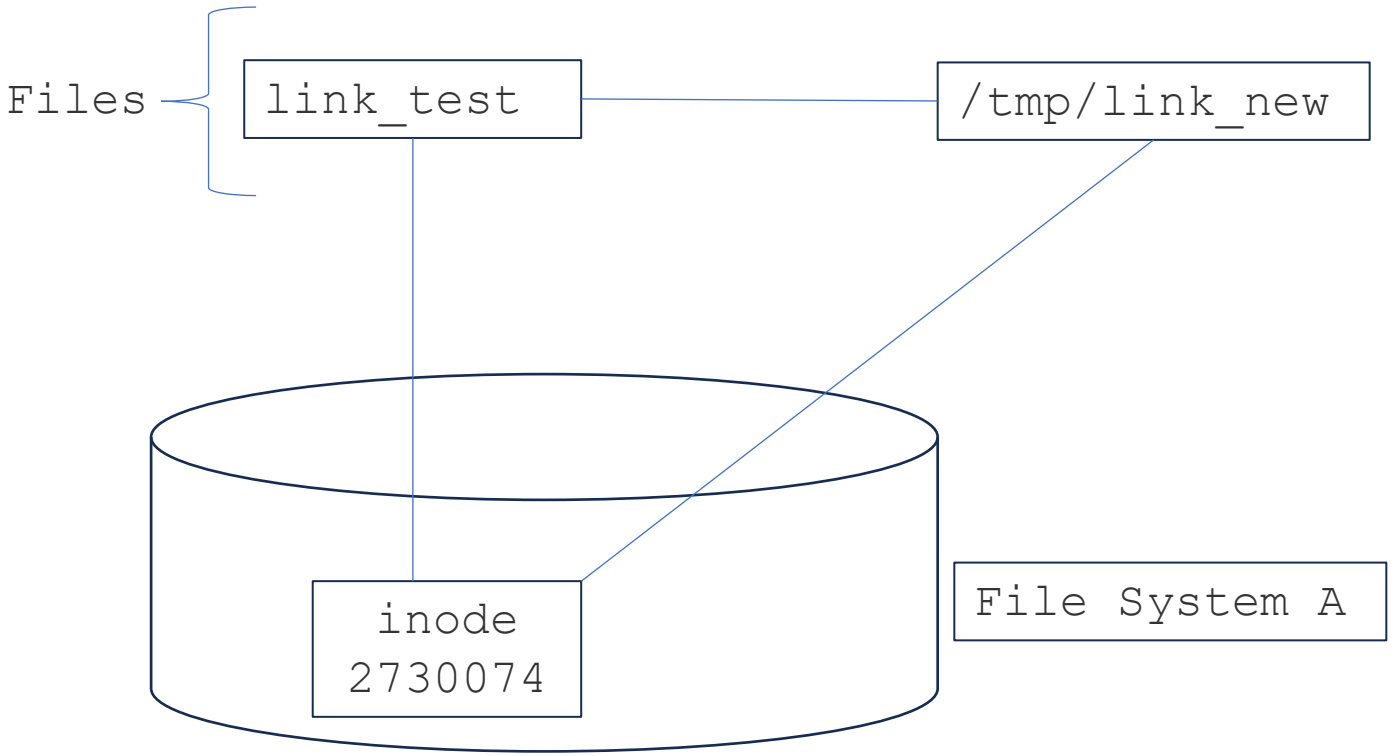
```
$ ln link_test /tmp/link_new

$ ls -l link_test /tmp/link_new
-rw-rw-r--  2  userA groupA 23 Jul 31 14:27 link_test
-rw-rw-r--  2  userA groupA 23 Jul 31 14:27 /tmp/link_new

ls -li link_test /tmp/link_new
2730074 -rw-rw-r-- 2 userA groupA 23 Jul 31 14:27 link_test
2730074 -rw-rw-r-- 2 tcarrigan tcarrigan 23 Jul 31 14:27 /tmp/link_new
```

The permissions, link count, ownership, timestamps, and file content are the exact same.

Hard links – number of links & inode



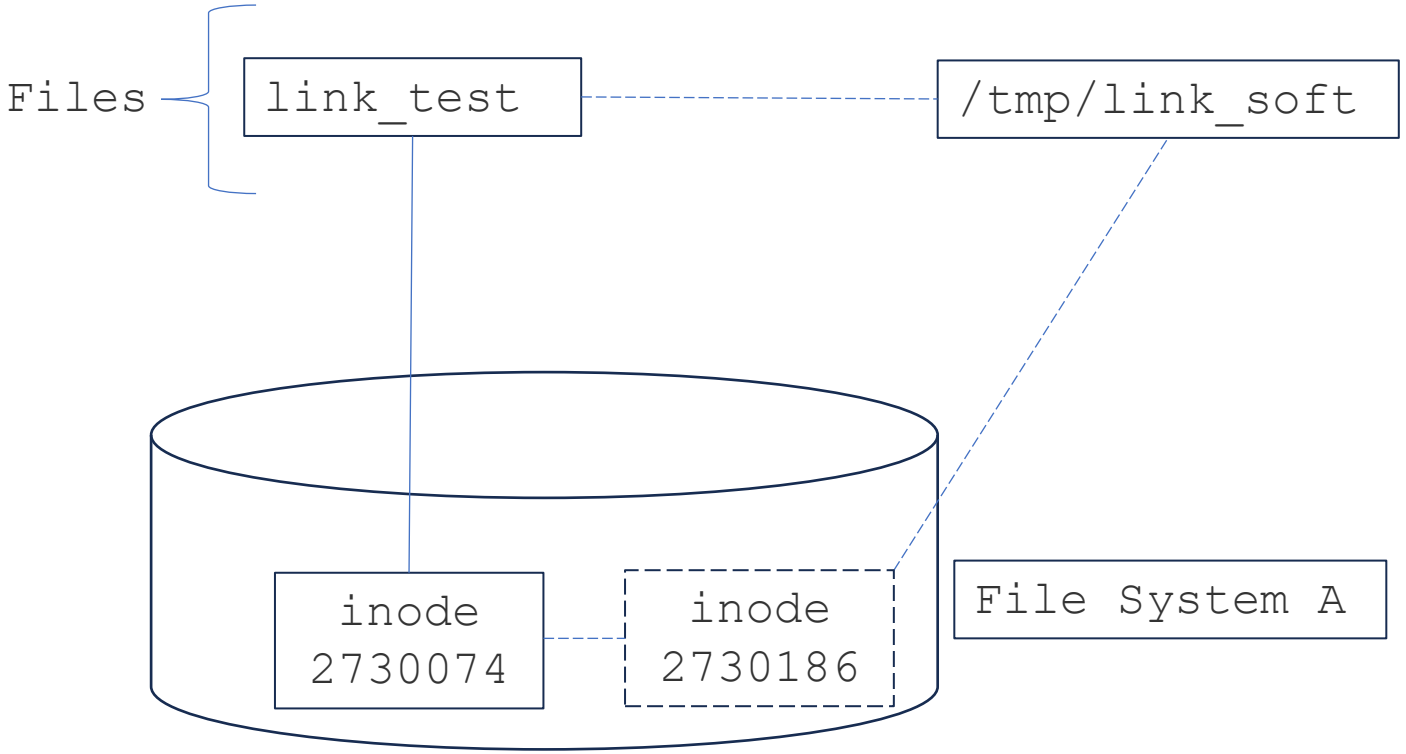
- When changes are made to one file, the other file reflects those changes.
- Hard links cannot be created across multiple file systems.
- If the original file (link_test) is deleted, the data still exists under the hard link (/tmp/link_new).
- The data is deleted from the file system when all links to the data are removed.

Soft Links

- Also known as symbolic links
- Command: ln –s <file1> <file2>
- Similar to shortcuts in Windows
- If the original file is deleted, the soft link is broken (also known as ‘dangling soft link’)
- Two different inode numbers
- Permissions and time stamps can be different

Soft links – number of links & inode

```
$ ls -s link_test /tmp/link_soft
```



- When changes are made to either of the files, both files reflect those changes.
- Soft links cannot be created across multiple file systems.
- If the original file (link_test) is deleted, the data does not exist under the soft link (/tmp/link_soft) but the soft link remains. Later if a new file 'link_test' is created, the soft link will point to it.
- The data is deleted from the file system when the main file is removed.

Differences between soft and hard links

Factor	Soft Link	Hard Link
inode	Soft link's inode number is different from the inode number of the original file or directory	Uses the same inode number of the original file
Linking of directories	Soft links can be used to link directories	Cannot link directories
Effect on deleting the original file	The soft link will not work. It is similar to 'shortcut' in Windows.	This does not delete the inode. The hard link will continue to work and show the file because it points to the same inode.
Memory consumption	More	Less
Speed or Performance	Slower than hard links	Faster than soft links

Summary

- Linux Commands – Basic Commands
- Working with Files and Folders, Changing Permissions & Ownership
- Types of Links – Soft Links and Hard Links

Thank You