Essential commands - 2

Presentated by Otabek Tursunboev

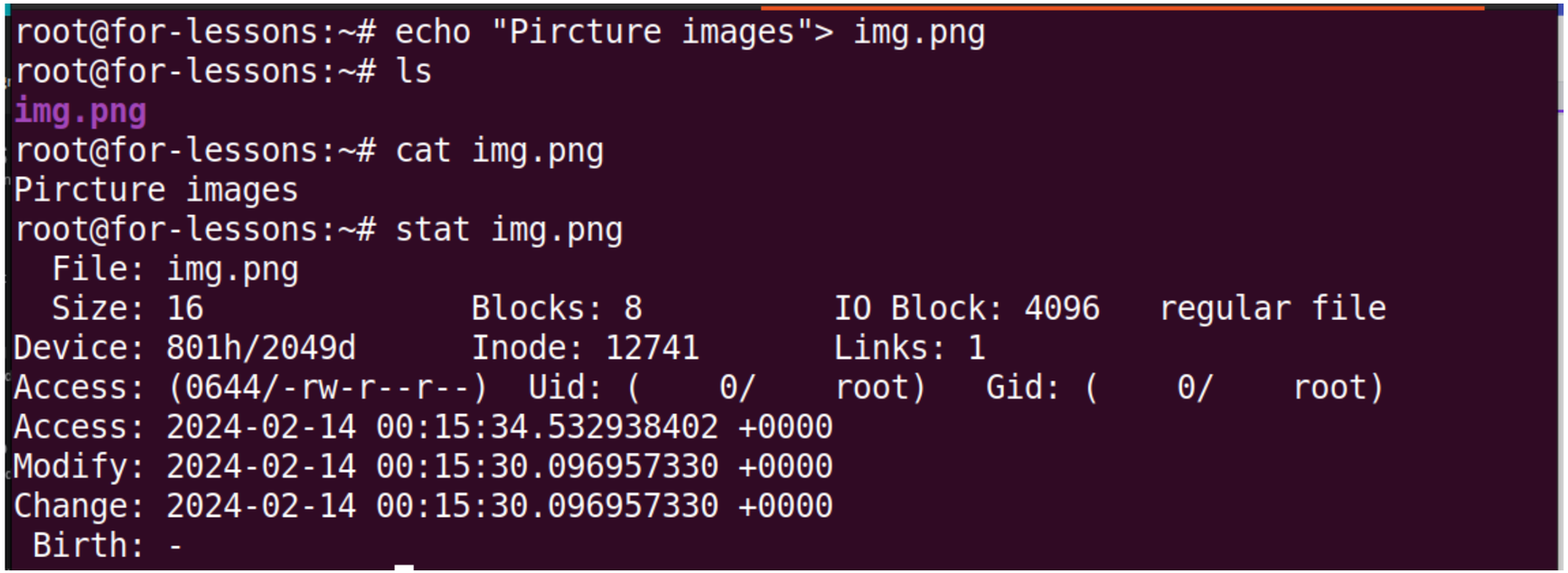
Inode

An inode is a concept in filesystems on Linux and Unix-like operating systems. Think of an inode as a sort of ID card for each file and directory on your computer. This ID card contains important information about the file or directory, but not the name or the actual data. Instead, it holds details like who owns the file (user and group), the file's permissions (who can read, write, or execute the file), the file's size, and where the file's actual data is stored on the disk.

When you create a file or directory, the system creates an inode for it and assigns it a unique number. This number is used by the system to reference the file or directory. However, when you want to access a file, you use its name, and the system uses a separate table to map the name to the inode number, so it knows which inode to look at for the details.

Imagine a library where every book has a unique code. The code itself doesn't tell you the title of the book or its contents, but it does tell the library's system where the book is located, who donated it, when it was added to the library, and more. The library's catalog, which you use to find books by title, matches each title to its unique code, similar to how file names are matched to inodes in the computer's filesystem.

So, in simple words, an inode is a data structure that stores all the metadata of a file or directory (except its name and actual content), including its permissions, owner, size, and where its data blocks are on the disk.


root@for-lessons:~# echo "Pircture images"> img.png
root@for-lessons:~# ls
img.png
root@for-lessons:~# cat img.png
Pircture images
root@for-lessons:~# stat img.png
File: img.png
Size: 16 Blocks: 8 IO Block: 4096 regular file
Device: 801h/2049d Inode: 12741 Links: 1
Access: (0644/-rw-r--r--) Uid: (0/ root) Gid: (0/ root)
Access: 2024-02-14 00:15:34.532938402 +0000
Modify: 2024-02-14 00:15:30.096957330 +0000
Change: 2024-02-14 00:15:30.096957330 +0000
Birth: -

Links

Hard links

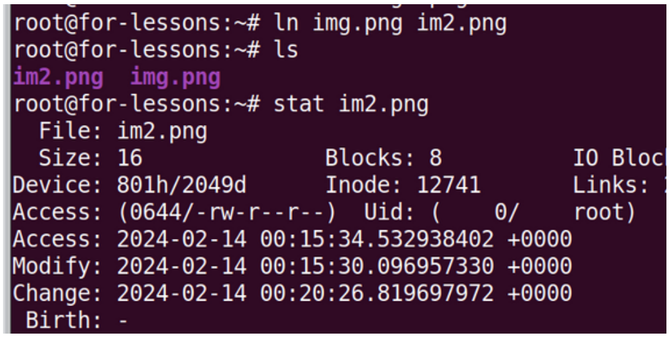
Soft links

Hard link

A hard link is another name for an existing file on the filesystem. When you create a hard link to a file, you're creating a new directory entry for the file, but both the original file and the hard link share the same inode. This means they are indistinguishable from each other at the data level; the system sees them as two entries for the same file.

Characteristics:

- Does not consume additional space for the file's data.
- Deleting the original file does not remove the data on the disk as long as a hard link to it exists. The data remains accessible through the hard link.
- Hard links cannot span across different filesystems, meaning you can't create a hard link to a file that's on a different disk or partition.
- You cannot create a hard link for directories to prevent creating loops within the filesystem.



Soft link

A symbolic link is a special type of file that points to another file or directory. It's like a shortcut in Windows or an alias on macOS. Unlike a hard link, a symbolic link does not share the same inode as the original file. Instead, it contains a text path to the linked file's location.

Characteristics:

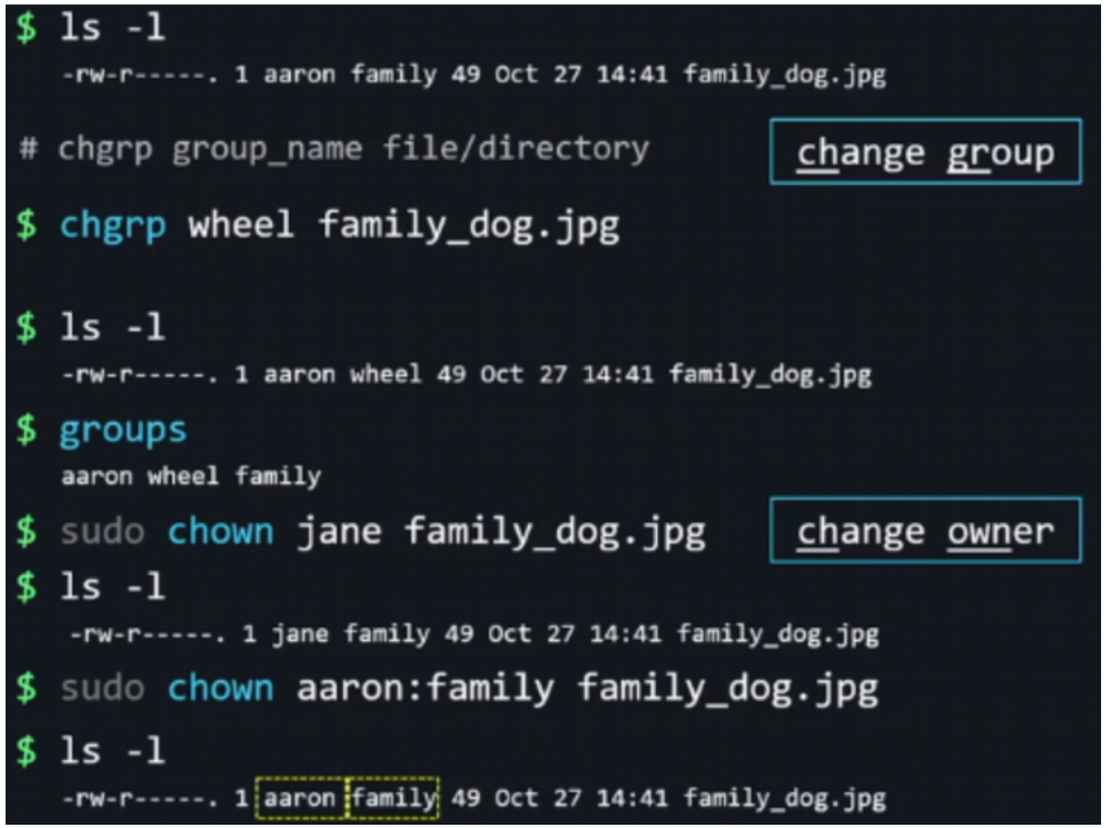
- Consumes a small amount of space to store the path information.
- If the original file is deleted, the symbolic link becomes a "dangling" link that points to a non-existent file, rendering it unusable until it's linked to a new file.
- Can span across different filesystems since it's essentially a pointer to a file path, not directly to the file's data.
- You can create symbolic links to directories, making them very useful for creating shortcuts or organizing files and directories flexibly.

```
root@for-lessons:~# ln -s im2.png img.png
root@for-lessons:~# ls
im2.png  img.png
root@for-lessons:~# stat im2.png
  File: im2.png
  Size: 16              Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 12741       Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2024-02-14 00:24:22.530700604 +0000
Modify: 2024-02-14 00:15:30.096957330 +0000
Change: 2024-02-14 00:24:18.830716253 +0000
 Birth: -
root@for-lessons:~# stat img.png
  File: img.png -> im2.png
  Size: 7               Blocks: 0          IO Block: 4096   symbolic link
Device: 801h/2049d      Inode: 12960       Links: 1
Access: (0777/lrwxrwxrwx)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2024-02-14 00:31:05.369015451 +0000
Modify: 2024-02-14 00:31:04.037020849 +0000
Change: 2024-02-14 00:31:04.037020849 +0000
 Birth: -
```

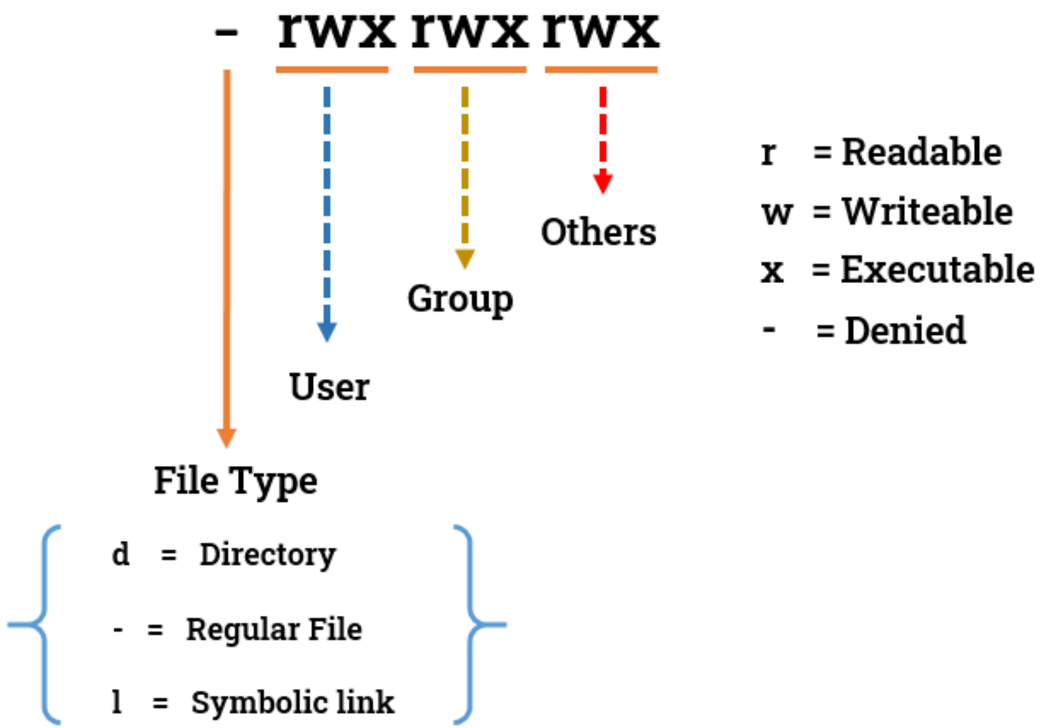
File types


$ ls -l
-rwxrwxrwx. 1 aaron family 49 Oct 27 14:41 family_dog.jpg
File Type
Identifier
DIRECTORY
d
REGULAR FILE
-
CHARACTER DEVICE
c
LINK
l
SOCKET FILE
s
PIPE
p
BLOCK DEVICE
b

File ownership



File and directory permissions



Accepting permissions

Add permissions

```
root@for-lessons:~# ls -l
total 4
-r--r--r-- 1 root root 21 Feb 14 00:40 im2.png
lrwxrwxrwx 1 root root  7 Feb 14 00:31 img.png -> im2.png
root@for-lessons:~# chmod u+wx im2.png
root@for-lessons:~# ls -l
total 4
-rwxr--r-- 1 root root 21 Feb 14 00:40 im2.png
lrwxrwxrwx 1 root root  7 Feb 14 00:31 img.png -> im2.png
root@for-lessons:~#
```

Change permissions

```
$ ls -l
-r--rw----. 1 aaron family 49 Oct 27 14:41 family_dog.jpg

$ chmod g=r family_dog.jpg
$ ls -l
-r--r-----. 1 aaron family 49 Oct 27 14:41 family_dog.jpg

$ chmod g=rw family_dog.jpg
$ ls -l
-r--rw----. 1 aaron family 49 Oct 27 14:41 family_dog.jpg

$ chmod g= family_dog.jpg
$ ls -l
-r--------. 1 aaron family 49 Oct 27 14:41 family_dog.jpg

$ chmod g-rwx family_dog.jpg
```

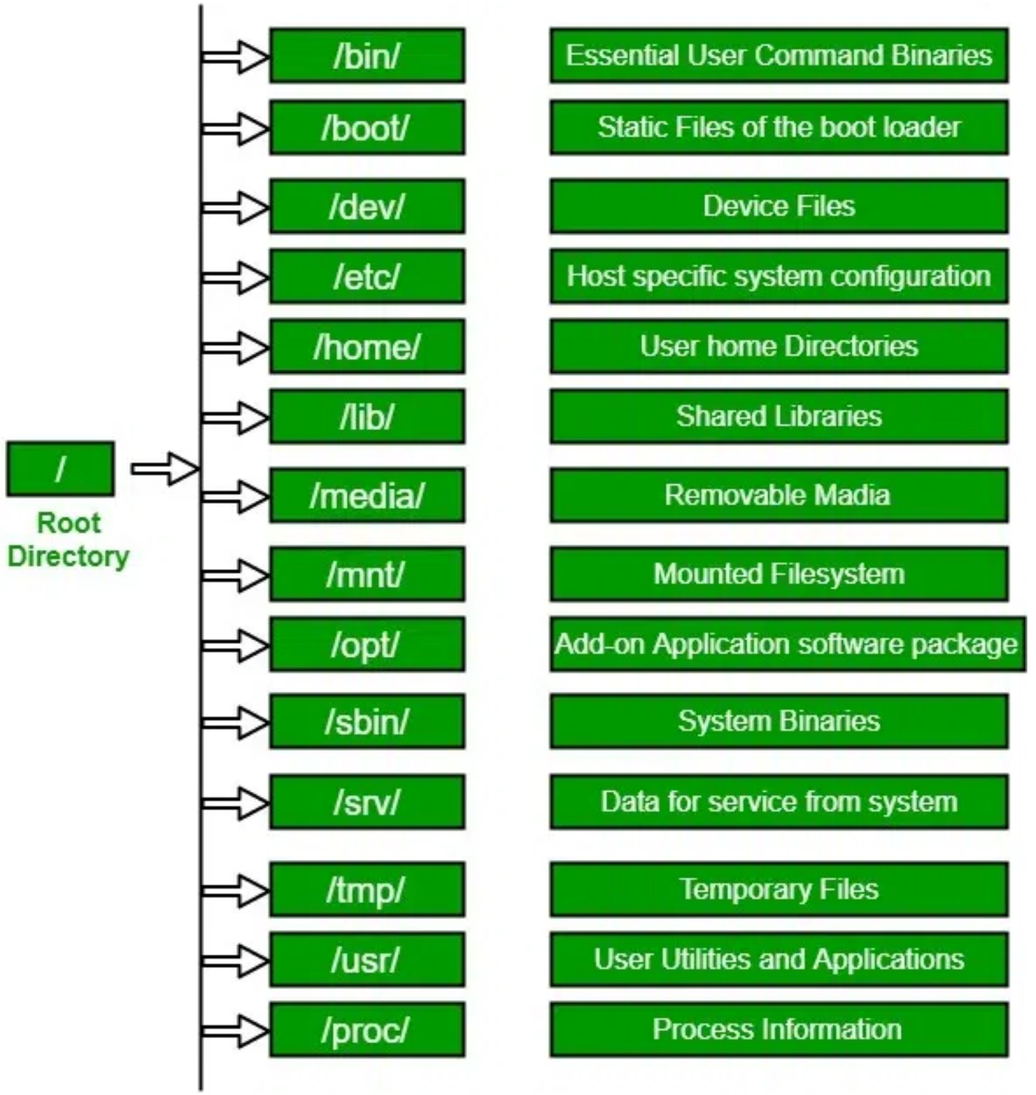
u=[list of permissions]

	Option	Examples
user	u=	u=w / u=rw / u=rwx
group	g=	g=w / g=rw / g=rwx
others	o=	o=w / o=rw / o=rwx

Permissions in octal

Octal Representation			
0	000	- - -	No permissions
1	001	- - x	Only Execute
2	010	- w -	Only Write
3	011	- w x	Write and Execute
4	100	r - -	Only Read
5	101	r - x	Read and Execute
6	110	r w -	Read and Write
7	111	r w x	Read, Write and Execute

Linux file system



- /bin: This directory contains essential command binaries that are needed for the system to boot and run in single-user mode.
- /boot: This directory contains boot-related files, such as the Linux kernel, the initramfs, and bootloader configuration files.
- /dev: This directory contains device files, which are special files that represent devices such as printers, keyboards, and disk drives.
- /etc: This directory contains system-wide configuration files and directories.
- /home: This directory contains the home directories of users on the system. Each user has a separate subdirectory under /home, where they can store their personal files and configurations.
- /lib: This directory contains library files that are needed by the command binaries in the /bin and /sbin directories.
- /root: This directory is the home directory of the “root” user, which is the superuser
- account with administrative privileges.
- /sbin: This directory contains essential system binaries, such as the init daemon and the
- reboot command.
- /tmp: This directory is a temporary storage area for files that are being used by
- applications. These files are usually deleted when the system is rebooted.
- /usr: This directory contains user-related files and directories, such as applications,
- libraries, and documentation.
- /var: This directory contains files and directories that are expected to change during the
- operation of the system, such as log files and temporary files.