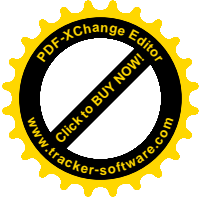


Columns of ls -l



```
ssoomm@system64U: /opt/skype-4.3.0.37$ ls -l
total 35740
drwxr-xr-x 6 ssoomm ssoomm 4096 feb 9 15:29 ./
drwxr-xr-x 4 root root 4096 feb 9 15:12 ../
drwxr-xr-x 2 ssoomm ssoomm 4096 may 22 2014 avatars/
drwxr-xr-x 2 ssoomm ssoomm 4096 may 22 2014 icons/
drwxr-xr-x 2 ssoomm ssoomm 4096 may 22 2014 lang/
-rw-r--r-- 1 ssoomm ssoomm 33513 may 22 2014 LICENSE
-rw-r--r-- 1 ssoomm ssoomm 9059 may 22 2014 README
-rw-r--r-- 1 ssoomm ssoomm 36499124 may 22 2014 skype*
-rw-r--r-- 1 ssoomm ssoomm 453 may 22 2014 skype.conf
-rwxr-xr-x 1 ssoomm ssoomm 231 feb 9 15:29 skype.desktop*
-rw-r--r-- 1 root root 1024 feb 9 15:29 .skype.desktop.swp
drwxr-xr-x 2 ssoomm ssoomm 4096 may 22 2014 sounds/
-rw-r--r-- 1 ssoomm ssoomm 10639 may 22 2014 third-party_attributions.txt
ssoomm@system64U: /opt/skype-4.3.0.37$ ./skype
bash: ./skype: No existe el archivo o el directorio
ssoomm@system64U: /opt/skype-4.3.0.37$
```

d indicates directory
and hyphen indicates
not a directory

owner of file

group

file size in bytes

indicates number of hard links
(explained below)

filename

date of last modifica-
tion (time included if yr
is current yr; otherwise
yr is included)

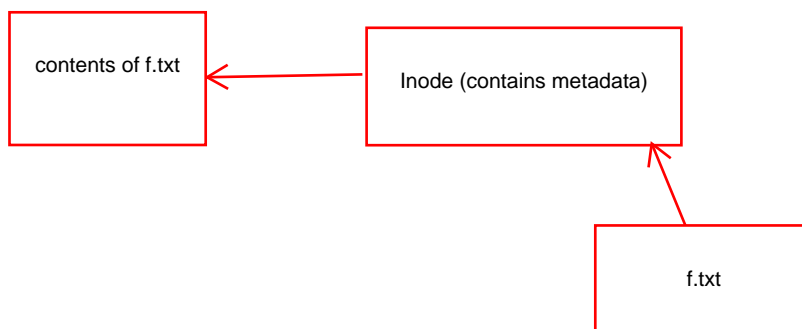
*The three segments of r,w,x (, or hyphen) indicate read, write and execute permissions for the file owner, group and everyone else (going from left to right); every type of user (whether a file owner, group, or everyone else) is designated 3 spaces for read(r), write(w) and execute(x) - in that order. If the user isn't authorized for that particular permission, a hyphen will appear in its place instead of that corresponding letter.

For example, in the bottom-most file in the listing (*third-party_attributions.txt*), the owner has read and write permissions, while the group and everyone else only have only read permissions. No one has execute permissions (which makes sense since it's a text file- not an executable)

Source of above image: <https://askubuntu.com/questions/732149/cant-execute-this-file> (visited 2/15/2018)

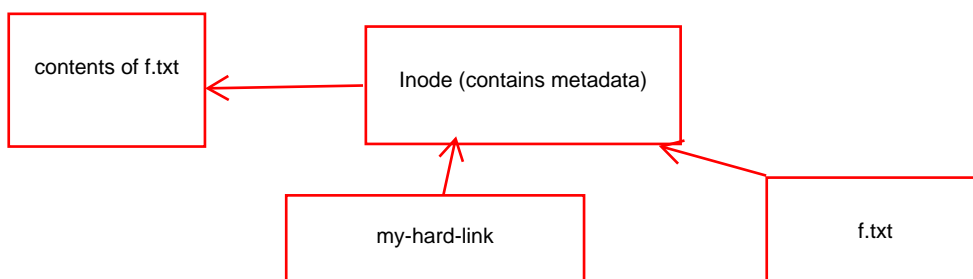
Hard links vs symbolic links (aka. soft links) - as I understand them

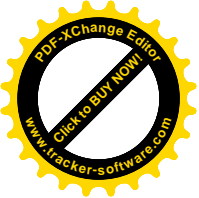
* When you create a file (say f.txt) with some content, the file system creates a new inode (not to be confused with a node) with metadata about the file and a pointer to the data; it also creates the hardlink f.txt; this hardlink is a pointer to the inode just created. (situation diagrammed below)



* Suppose we ran the below in bash
\$ ln f.txt my-hard-link.

We've now created another hard link to f.txt (i.e. we've created another pointer to the Inode containing the metadata about the contents of f.txt- the pointer's name is my-hard-link. (situation diagrammed below)





* Now we'll create a *soft link* to f.txt. You'll see that this means that this means we'll be creating a pointer to the filename object (which is a pointer to the Inode- which is in turn a pointer to the file contents)- so we'll be yet another level of indirection removed from the file contents relative to what we'd be had we just created a hard link.
\$ ln -s f.txt my-soft-link

