Inodes and Linking

Inodes: An inode (index node) is a data structure on a traditional Unix-style file system. It stores basic information about a file, directory, or other file system object. Each object in the filesystem is associated with an inode.

Each and every file under Linux (and UNIX) has the following attributes (metadata) stored in an inode:
=> File type (executable, block special etc)
=> Permissions (read, write etc)
=> Owner
=> Group
=> File Size
=> File access, change and modification time (remember UNIX or Linux never stores file creation time, this is favorite question asked in UNIX/Linux sys admin job interview)
=> File deletion time
=> Number of links (soft/hard)
=> Extended attribute such as append only or no one can delete file including root user (immutability)
=> Access Control List (ACLs)

Perhaps most importantly, each inode stores the disk block location(s) of the object's data. This is the connection between the file within the filesystem and the actual data that it represents.

Each inode has a unique inode number.
You can use `ls –i` to see the inode number of a file:

```
ls –i test.txt
1839562 test.txt
```

These numbers index into a table of inodes (maintained by the OS) in a known location on the device. From there, the kernel's file system driver can access the inode contents, including the location of the file - thus allowing access to the file's data wherever it is stored.

You can also use the `stat` command to see the inode number and attributes of a file.

If you have a filename that includes special characters like ? or *, you might need to refer to it with the inode number in commands.

Directories have inodes just like files.

Links

There are two types of links: hard links and soft (symbolic) links.

Hard Links: With hard links, it is possible to associate multiple filesystem entities with a single inode. To create a hard link use the `ln` command as follows:

```
ln ./file1 ./file2
```

Here, file1 must be an existing filesystem entity. This will create file2 which will refer to the same inode as file1. This means that they both point to the same data. Editing file2 is the same as editing file1.

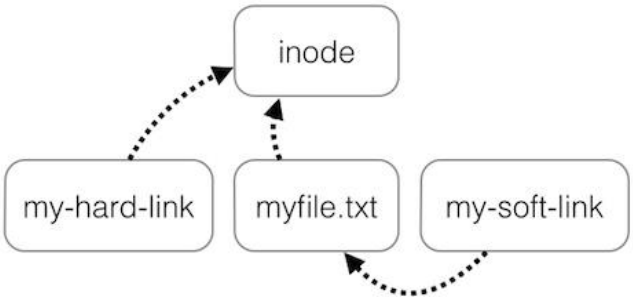
Hard links always refer to the inode of the source, even if the source is moved or removed. In other words, moving or removing file1 will not change the behavior of file2 at all. The data will still be accessible.

Hard links cannot link directories. They also cannot cross filesystem boundaries.

Soft Links: Instead of linking to an inode, soft links refer to another link. We can create soft links by with `ln –s` as shown in the example below:

```
ln -s ./file1 ./file2
```

In the example above, file2 points to the link file1, which in turn points to the inode specifying the location of the data on disk. This picture shows the difference between soft and hard links.



Soft links can link directories and cross filesystem boundaries (so a symbolic link to data on one drive or partition can exist on another drive or partition). However, if the source of the link is changed or removed, the soft link is not updated and will be broken.