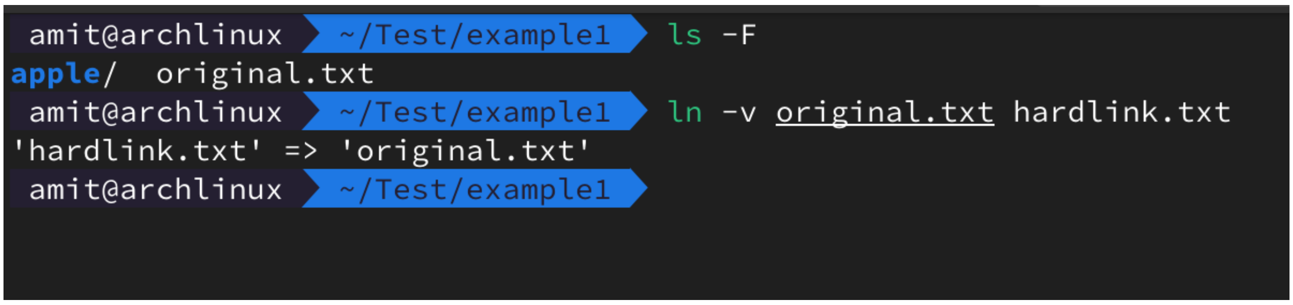
Hard Link

A hard link is a reference to the same inode of a file in a filesystem. It creates an additional file entry that points to the exact same data blocks on the disk as the original file. As a result, hard links allow multiple filenames to refer to the same file content, without duplicating the actual data.

Creating Hard Link:

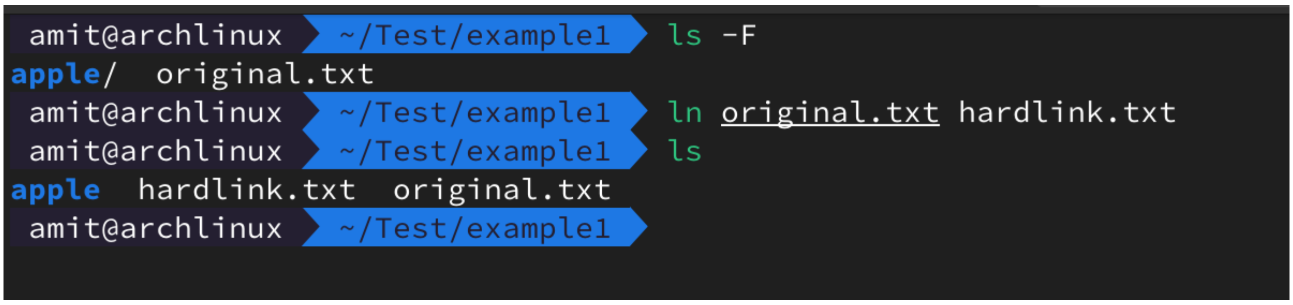
```
$ ln [option]  target link_name
```

example:



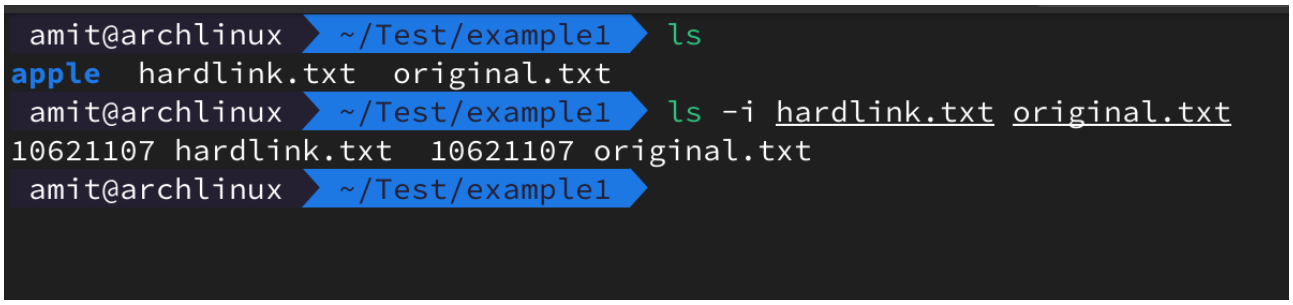
```
 amit@archlinux  ~/Test/example1  ls -F
apple/  original.txt
 amit@archlinux  ~/Test/example1  ln -v original.txt hardlink.txt
'hardlink.txt' => 'original.txt'
 amit@archlinux  ~/Test/example1 
```

or



```
 amit@archlinux  ~/Test/example1  ls -F
apple/  original.txt
 amit@archlinux  ~/Test/example1  ln original.txt hardlink.txt
 amit@archlinux  ~/Test/example1  ls
apple  hardlink.txt  original.txt
 amit@archlinux  ~/Test/example1 
```

They share same inode number.



```
 amit@archlinux  ~/Test/example1  ls
apple  hardlink.txt  original.txt
 amit@archlinux  ~/Test/example1  ls -i hardlink.txt original.txt
10621107 hardlink.txt  10621107 original.txt
 amit@archlinux  ~/Test/example1 
```

Advantages of Hard Link

1. Efficient Use of Disk Space

- **No Data Duplication:** Hard links allow multiple files to reference the same data on the disk without duplicating the actual content. This means you can have multiple access points to a file, but the data is stored only once. This can be particularly useful for saving disk space when you need to reference large files in different locations.

2. Data Redundancy

- **Increased Reliability:** Since hard links are independent of the original file name, the deletion of one hard link does not affect the others. As long as at least one hard link to the data exists, the data remains intact. This makes it harder to accidentally lose data.
- **Persistence:** When the original file is deleted, the data is not removed from the disk until all hard links are deleted. This makes hard links useful for ensuring data persistence until it's no longer needed.

3. File Synchronization

- **Unified Data Across Multiple Locations:** Changes made to the file content through any hard link are reflected in all hard links because they point to the same data on disk. This ensures that multiple users or processes can access the same file data without inconsistency.

4. Improved Backup and Versioning

- **Space-Efficient Backup:** Hard links can be used for creating backups without duplicating data. For example, you could create a new hard link for each version of a file. Even though each version appears as a separate file, they all reference the same underlying data, avoiding unnecessary duplication of large files.
- **Snapshotting:** Hard links can be used in backup systems to create snapshot-like copies of files without actually duplicating the file data. This is useful in scenarios where space is limited but you still need to maintain different versions of files.

5. File Integrity and Data Recovery

- **Minimizing Data Loss Risk:** Since multiple hard links point to the same data, the risk of losing the file data is minimized if one link is lost (e.g., if one link is deleted by mistake).
- **Recovery of Lost Files:** Even if a file is deleted, its data may still be accessible via the remaining hard link(s), allowing recovery of important data.

6. No Dependency on Path

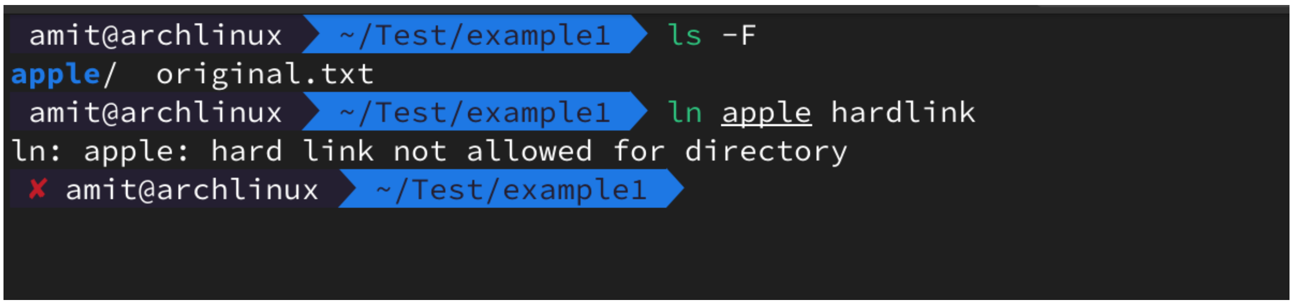
- **Less Vulnerable to Path Changes:** Hard links are not dependent on the path or location of the file. If a file is moved to a different directory, all hard links to that file still work because they are based on the inode (the actual data address on disk), not the file path. This contrasts with soft links, which may break if the original file is moved.

Limitations and Considerations:

While hard links provide several benefits, there are also limitations:

- **Cannot Link Directories:** Hard links cannot be used for directories (except for the special `.` and `..` directories), which prevents them from being used to create complex directory structures.

  Example:

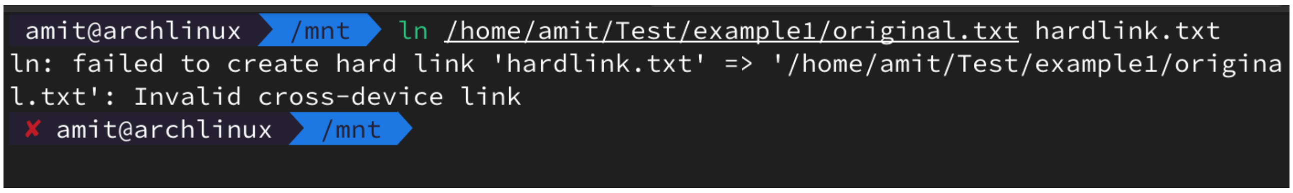


```
 amit@archlinux  ~/Test/example1  ls -F
apple/  original.txt
 amit@archlinux  ~/Test/example1  ln apple hardlink
ln: apple: hard link not allowed for directory
 ✘ amit@archlinux  ~/Test/example1
```

Why this restriction exists ?

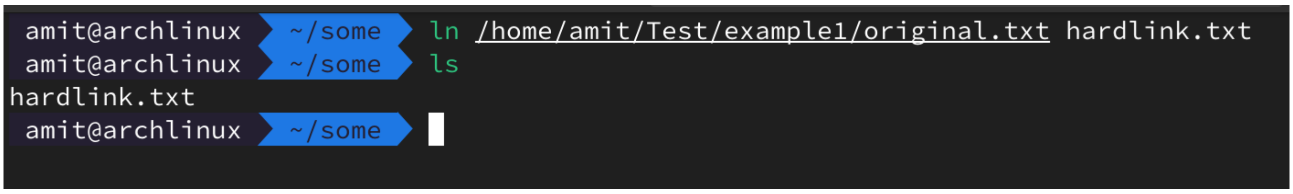
- **Circular References:** If you could create hard links for directories, it would be possible to create circular directory structures. For example, if `dir1` is linked to `dir2`, and `dir2` is linked to `dir1`, it would create an infinite loop when trying to traverse the directories.
- **Filesystem Integrity:** Allowing hard links for directories could break the filesystem's structure and make operations like file deletion, directory traversal, and inode management very complex and error-prone.

- **Not Portable Across File Systems:** Hard links work within the same file system. They cannot span across different file systems (e.g., between different partitions or disks). Since hard links reference the same inode, they must exist within the same filesystem, because **inodes are specific to a filesystem**. Different filesystems (e.g., different partitions, disks, or mounted volumes) have independent inode tables. As a result, hard links cannot span across different filesystems, because **inodes from one filesystem cannot be referenced by a different filesystem.**



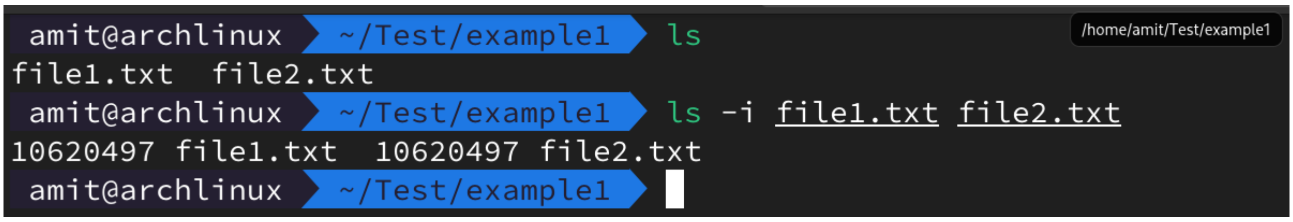
```
 amit@archlinux  /mnt  ln /home/amit/Test/example1/original.txt hardlink.txt
ln: failed to create hard link 'hardlink.txt' => '/home/amit/Test/example1/origina
l.txt': Invalid cross-device link
 ✘ amit@archlinux  /mnt
```

But works everywhere in same filesystem.



```
 amit@archlinux  ~/some  ln /home/amit/Test/example1/original.txt hardlink.txt
 amit@archlinux  ~/some  ls
hardlink.txt
 amit@archlinux  ~/some  █
```

- **Not Easily Identified:** Hard links can be more difficult to manage or track compared to soft links because they appear as regular files with no visible indication that they are hard links, unless you check their inode numbers.



```
 amit@archlinux  ~/Test/example1  ls                         /home/amit/Test/example1
file1.txt  file2.txt
 amit@archlinux  ~/Test/example1  ls -i file1.txt file2.txt
10620497 file1.txt  10620497 file2.txt
 amit@archlinux  ~/Test/example1 
```