Hard Link vs Soft Link in Linux

In Linux, hard links and soft (symbolic) links are used to reference files in different ways.

This document compares both types of links with examples and diagrams for better understanding.

Comparison Overview

Hard Link:

- Direct pointer to file's inode

- Same inode number as original

- Changes in one reflect in all

- Cannot link to directories

- Cannot span across filesystems

- Survives if original is deleted

Command: ln original.txt hardlink.txt

Soft Link (Symbolic Link):

- Pointer to filename

- Different inode

- Becomes broken if original is deleted

- Can link to directories

- Can span across filesystems

Command: ln -s original.txt softlink.txt

Hands-On Example

Hard Link vs Soft Link in Linux

Step-by-Step Example:

1. Create a file:

```
echo "This is the original file" > original.txt
```

2. Create a hard and soft link:

```
ln original.txt hardlink.txt
ln -s original.txt softlink.txt
```

3. Check inodes:

```
ls -li original.txt hardlink.txt softlink.txt
```

4. Modify content via hardlink:

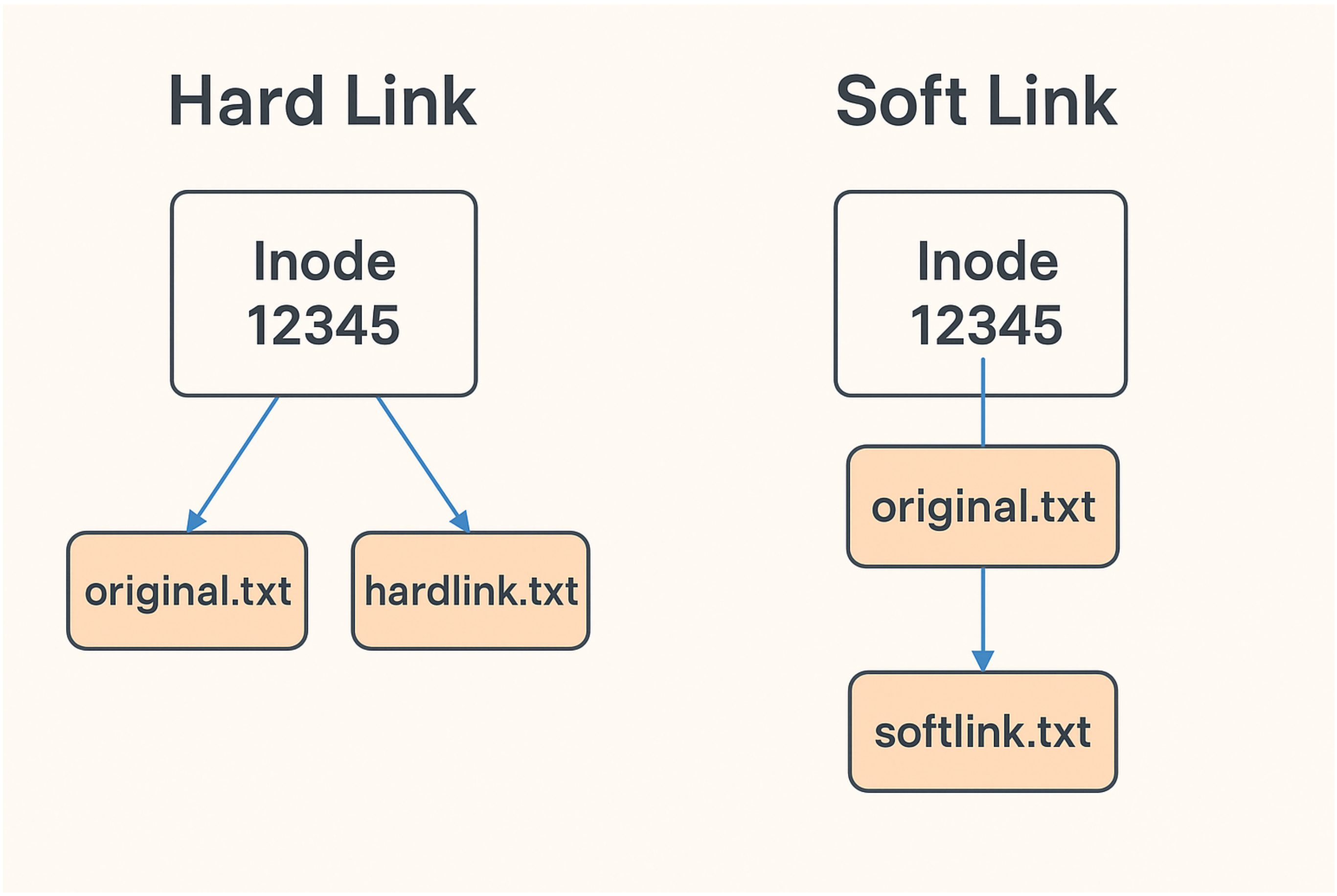
```
echo "Appended line" >> hardlink.txt
```

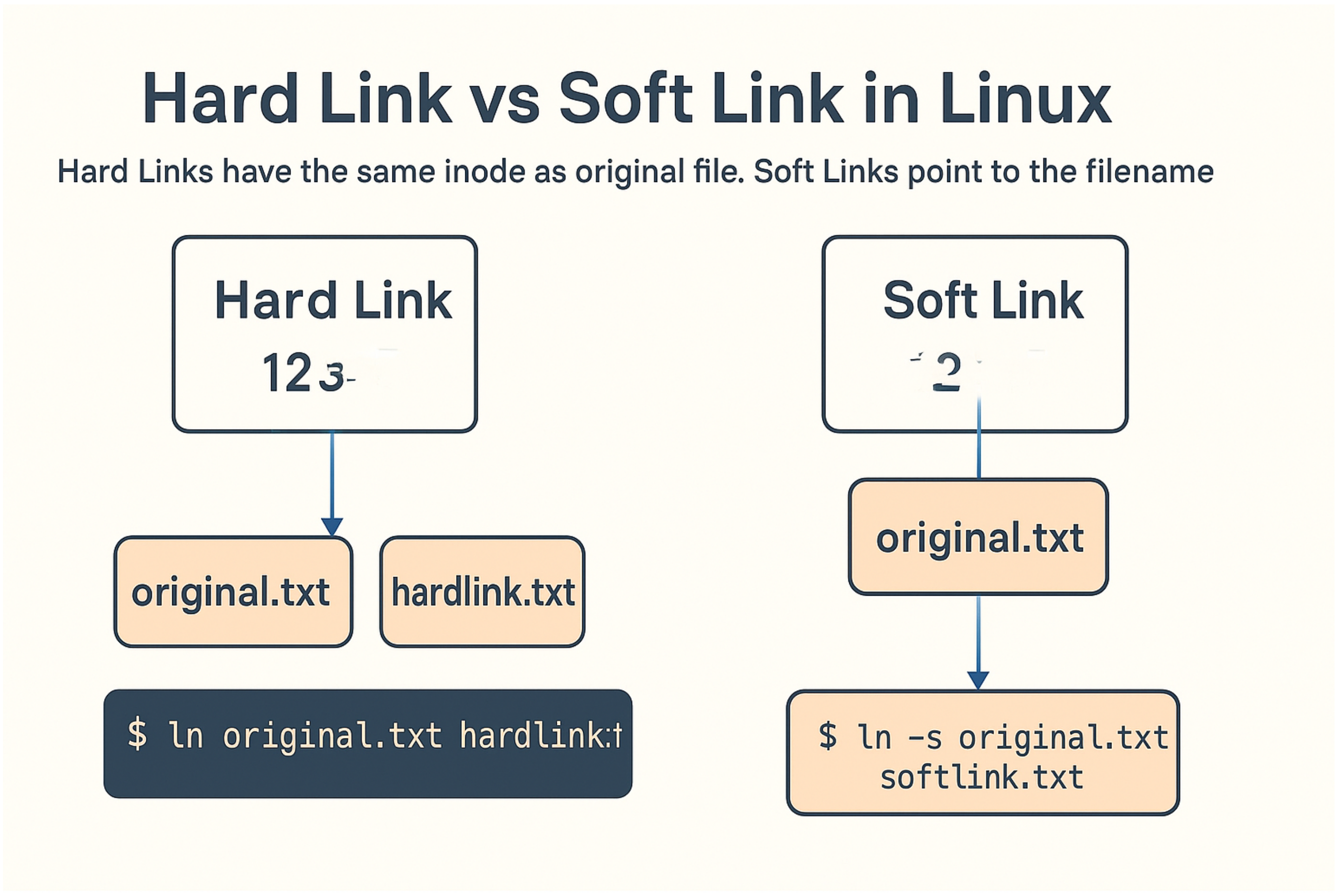
5. Delete original and observe behavior:

```
rm original.txt
cat hardlink.txt  # still works
cat softlink.txt  # broken link
```

Visual Diagrams

Hard Link vs Soft Link in Linux





Hard Links have the same inode as original file. Soft Links point to the filename

```
$ ln original.txt hardlink.t
```

```
$ ln -s original.txt
softlink.txt
```

Hard Link vs Soft Link in Linux

Document authored by: Ekansh Gupta

Generated on: 2025-06-13