**Inode**

# Inode

The inode is a data structure in a Unix-style file system that describes a filesystem object such as a file or a directory. Each inode stores the attributes and disk block location(s) of the object's data.

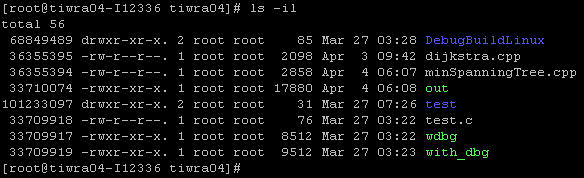
Filesystem object attributes may include metadata (times of last change, access, modification), as well as owner and permission data.

Directories are lists of names assigned to inodes. A directory contains an entry for itself, its parent, and each of its children.

A file system relies on data structures about the files, beside the file content. The former is called metadata—data that describe data. Each file is associated with an inode, which is identified by an integer number, often referred to as an i-number or inode number.

Inodes store information about files and directories (folders), such as file ownership, access mode (read, write, execute permissions), and file type. On many types of file system implementations, the maximum number of inodes is fixed at file system creation, limiting the maximum number of files the file system can hold.

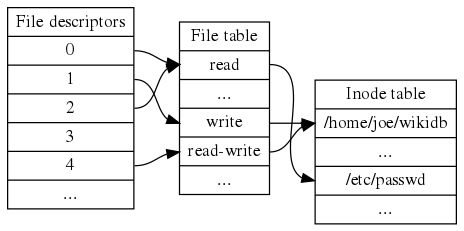
A file's inode number can be found using the ls -i command. The ls -i command prints the i-node number in the first column of the report.



File names and directory implications:

* Inodes do not contain its hardlink names, only other file metadata.
* Unix directories are lists of association structures, each of which contains one filename and one inode number.
* The file system driver must search a directory looking for a particular filename and then convert the filename to the correct corresponding inode number.

The operating system kernel's in-memory representation of this data is called **struct inode** in Linux.



# Hard Link

A hard link is a directory entry that associates a name with a file on a file system.

Creating a hard link has the effect of giving one file multiple names (e.g. different names in different directories) all of which independently connect to the same data on the disk, none of which depends on any of the others

Support also depends on the type of file system being used. For instance, the NTFS file system supports hard links, while FAT and ReFS do not.

On POSIX-compliant and partially POSIX-compliant operating systems, such as all Unix-like systems, additional hard links to existing files are created with the link() system call, or the ln and link command-line utilities. The stat command can reveal how many hard links point to a given file. The link count is also included in the output of ls -l.

On Microsoft Windows, hard links can be created using the mklink /H command on Windows NT 6.0 and later systems.

The process of unlinking dissociates a name from the data on the volume without destroying the associated data. The data is still accessible, as long as at least one link that points to it still exists. When the last link is removed, the space is considered free.

## Link counter

Most file systems that support hard links use reference counting. An integer value represents the total number of links that have been created to point to the data. When a new link is created, this value is increased by one. When a link is removed, the value is decreased by one. If the link count becomes zero, the operating system usually automatically deallocates the data space of the file if no process has the file opened for access, but it may choose not to do so immediately, either for performance sake or to enable undelete command.

On POSIX-compliant operating systems, such as many Unix-variants, the reference count for a file or directory is returned by the **stat()** or **fstat()** system calls in the **st\_nlink** field of **struct stat**.

## Limitations of hard links

1. To prevent loops in the filesystem, and to keep the interpretation of .. (parent directory) consistent, many modern operating systems do not allow hard links to directories.
2. Hard links can be created to files only on the same volume. If a link to a file on a different volume is needed, it may be created with a symbolic link.
3. The maximum number of hard links to a single file is limited by the size of the reference counter. On Unix-like systems the counter is usually machine-word-sized (32- or 64-bit: 4,294,967,295 or 18,446,744,073,709,551,615 links, respectively).
4. Though in some filesystems the number of hard links is limited more strictly by their on-disk format. As of Linux 3.11, the ext4 filesystem limits the number of hard links on a file to 65,000. Windows with NTFS filesystem has a limit of 1024 hard links on a file.

# Symbolic Link

A symbolic link (also symlink or soft link) is a term for any file that contains a reference to another file or directory in the form of an absolute or relative path and that affects pathname resolution.

If a symbolic link points to a target, and sometime later that target is moved, renamed or deleted, the symbolic link is not automatically updated or deleted, but continues to exist and still points to the old target, now a non-existing location or file. Symbolic links pointing to moved or non-existing targets are sometimes called broken, orphaned, dead, or dangling.

Symbolic links are different from hard links. Hard links do not link paths on different volumes or file systems, whereas symbolic links may point to any file or directory irrespective of the volumes on which the link and target reside. Hard links always refer to an existing file, whereas symbolic links may contain an arbitrary path that does not point to anything.

