

inode

In a **Unix-style file system**, the **inode** is a **data structure** used to represent a filesystem object, which can be one of various things including a file or a directory. Each inode stores the attributes and disk block location(s) of the filesystem object's data.^[1] Filesystem object attributes may include manipulation **metadata** (e.g. change,^[2] access, modify time), as well as owner and **permission** data (e.g. group-id, user-id, permissions).^[3]

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

1 Etymology

There has been some uncertainty on the **Linux kernel mailing list** as to the reason for designating these as “i” nodes. The question was brought to Unix pioneer **Dennis Ritchie**, who replied:^[4]

In truth, I don't know either. It was just a term that we started to use. “Index” is my best guess, because of the slightly unusual file system structure that stored the access information of files as a flat array on the disk, with all the hierarchical directory information living aside from this. Thus the i-number is an index in this array, the i-node is the selected element of the array. (The “i-” notation was used in the 1st edition manual; its hyphen was gradually dropped.)

A paper published in 1978 by Ritchie and **Ken Thompson** strongly suggests the same etymology:^[5]

As mentioned in Section 3.2 above, a directory entry contains only a name for the associated file and a pointer to the file itself. This pointer is an integer called the *i-number* (for index number) of the file. When the file is accessed, its i-number is used as an index into a system table (the *i-list*) stored in a known part of the device on which the directory resides. The entry found thereby (the file's *i-node*) contains the description of the file:...

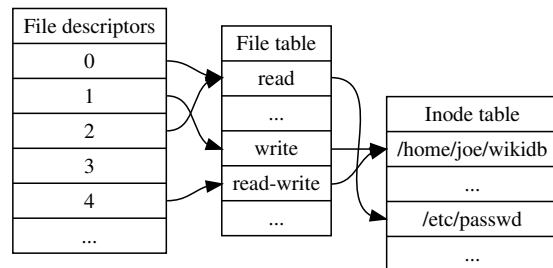
— The UNIX Time-Sharing System, The Bell System Technical Journal, 1978

Also, Maurice J. Bach writes:^[6]

The term inode is a contraction of the term index node and is commonly used in literature on the UNIX system.

— Maurice J. Bach, The Design of the Unix Operating System, 1986

2 Details



File descriptors, file table and inode table in Unix. Inspired by Maurice J. Bach (1986). The Design of the UNIX Operating System. Prentice Hall, p. 94.

A file system relies on data structures *about* the files, beside the file content. The former is called **metadata**—data that describes 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 typical allocation heuristic for inodes in a file system is one percent of total size.

The inode number indexes a table of inodes in a known location on the device; from the inode number, the file system driver portion of the kernel can access the contents of the inode, including the location of the file allowing access to the file.

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

Some Unix-style file systems such as **ReiserFS** omit an in-

ode table, but must store equivalent data in order to provide equivalent capabilities. The data may be called stat data, in reference to the `stat` system call that provides the data to programs.

File names and directory implications:

- inodes do not contain file 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. Systems derived from BSD use the term vnode, with the `v` of `vnode` referring to the kernel's virtual file system layer.

3 POSIX inode description

The POSIX standard mandates filesystem behavior that is strongly influenced by traditional UNIX filesystems. Regular files must have the following attributes:

- The size of the file in bytes.
- Device ID (this identifies the device containing the file).
- The User ID of the file's owner.
- The Group ID of the file.
- The file mode which determines the file type and how the file's owner, its group, and others can access the file.
- Additional system and user flags to further protect the file (limit its use and modification).
- Timestamps telling when the inode itself was last modified (ctime, *inode change time*), the file content last modified (mtime, *modification time*), and last accessed (atime, *access time*).
- A link count telling how many hard links point to the inode.
- Pointers to the disk blocks that store the file's contents (see inode pointer structure).

The `stat` system call retrieves a file's inode number and some of the information in the inode.

4 Implications

- Files can have multiple names. If multiple names hard link to the same inode then the names are equivalent; i.e., the first to be created has no special status. This is unlike symbolic links, which depend on the original name, not the inode (number).
- An inode may have no links. An unlinked file is removed from disk, and its resources are freed for re-allocation but deletion must wait until all processes that have opened it finish accessing it. This includes executable files which are implicitly held open by the processes executing them.
- It is typically not possible to map from an open file to the filename that was used to open it. The operating system immediately converts the filename to an inode number then discards the filename. This means that the `getcwd()` and `getwd()` library functions search the parent directory to find a file with an inode matching the working directory, then search that directory's parent, and so on until reaching the root directory. SVR4 and Linux systems maintain extra information to make this possible.
- Historically, it was possible to hard link directories. This made the directory structure into an arbitrary directed graph as opposed to a directed acyclic graph (DAG). It was even possible for a directory to be its own parent. Modern systems generally prohibit this confusing state, except that the parent of *root* is still defined as root. The most notable exception to this prohibition is found in Mac OS X (versions 10.5 and higher) which allows hard links of directories to be created by the superuser. This is used exclusively by Time Machine, Apple's full-system incremental backup utility.
- A file's inode number stays the same when it is moved to another directory on the same device, or when the disk is defragmented which may change its physical location. This also implies that completely conforming inode behavior is impossible to implement with many non-Unix file systems, such as FAT and its descendants, which don't have a way of storing this invariance when both a file's directory entry and its data are moved around.
- Installation of new libraries is simple with inode filesystems. A running process can access a library file while another process replaces that file, creating a new inode, and an all new mapping will exist for the new file so that subsequent attempts to access the library get the new version. This facility eliminates the need to reboot to replace currently mapped libraries. For this reason, when updating programs, best practice is to delete the old executable first and create a new inode for the updated version, so that

any processes executing the old version may proceed undisturbed.

5 Practical considerations

Many computer programs used by system administrators in Unix-like operating systems designate files with inode numbers. Examples include popular disk integrity checking utilities such as the `fsck` or `pfiles`. Thus, the need naturally arises to translate inode numbers to file pathnames and vice versa. This can be accomplished using the file finding utility `find` with the `-inum` option, or the `ls` command with the proper option (`-i` on POSIX-compliant platforms).

It is possible for a device to run out of inodes. When this happens, new files cannot be created on the device, even though there may be free space available. This is most common for use cases like mail servers which contain many small files.

Filesystems (such as `JFS` or `XFS`) escape this limitation with `extents` and/or dynamic inode allocation, which can 'grow' the filesystem and/or increase the number of inodes.

6 Inlining

It can make sense to store very small files in the inode itself to save both space (no data block needed) and look-up time (no further disk access needed). This file system feature is called inlining. The strict separation of inode and file data thus can no longer be assumed when using modern file systems.

If the data of a file fits in the space allocated for pointers to the data, this space can conveniently be used. For example, `ext2` and its successors stores the data of symlinks (typically file names) in this way, if the data is no more than 60 bytes ("fast symbolic links").^[7]

`Ext4` has a file system option called `inline_data` that, when enabled during file system creation, allows `ext4` to perform inlining. As an inode's size is limited, this only works for very small files.^[8]

7 See also

- `inode pointer structure`
- `inotify`

8 References

- [1] Andrew S. Tanenbaum, *Modern Operating Systems*, 3rd edition, page 279

- [2] JVSANTEN. "Difference between `mtime`, `ctime` and `atime` - Linux Howtos and FAQs". *Linux Howtos and FAQs - tutorials, guides and news for the Linux admins* - www.linux-faqs.info.
- [3] "Anatomy of the Linux virtual file system switch". *ibm.com*.
- [4] *Linux Kernel list archive*. Retrieved on 2011-01-12.
- [5] Ritchie, D.M.; Thompson, K. (1978). "The UNIX Time-Sharing System". *The Bell System Technical Journal* **57** (6): 1913–1914. Retrieved 19 December 2015.
- [6] "The Design of the UNIX Operating System: Maurice J. Bach: 0076092031369: Amazon.com: Books". *amazon.com*.
- [7] "The Linux kernel: Filesystems". *tue.nl*.
- [8] "Ext4 Disk Layout". *kernel.org*. Retrieved August 18, 2013.

9 External links

- *Anatomy Of The Linux File System*
- *Inode definition*
- *Explanation of Inodes, Symlinks, and Hardlinks*

10 Text and image sources, contributors, and licenses

10.1 Text

- **Inode** *Source:* <https://en.wikipedia.org/wiki/Inode?oldid=707367562> *Contributors:* Leandrod, Dominus, DIG~enwiki, TakuyaMurata, Delirium, Msablic, IMSoP, Furrykef, Bevo, Joy, AnonMoos, Ashley Y, David Gerard, Cedars, Laudaka, Digital infinity, Ryanaxp, Sam Hocevar, Eric B. and Rakim, GreenReaper, Abdull, David Strauss, Schuetzm, Dolda2000, Chungy, Giraffedata, Celada, Jumbuck, Andrewpmk, Seans Potato Business, Cdc, Voxadam, Popefelix, Kbolino, Aaron McDaid, Jenrzzz, AlbertCahalan~enwiki, Eyreland, Magister Mathematicae, Qwertyus, Crazyvas, Amoskahiga, Margosbot~enwiki, Nivix, Chobot, YurikBot, Steeltoe, Groogle, Brian1975, ScottyWZ, EAdherhold, Petef~enwiki, Singingwolfboy, Aeosynth, SmackBot, Mmernex, Rex the first, BiT, Brianski, EncMstr, Bazonka, Nbarth, Frap, Chlewbob, Sspecter, Emre D., Cybercobra, UDHA, Nigelm, Esoth~enwiki, Rorx, Abraxa~enwiki, Bezenek, RobinMiller, Shamsnaved, JoeBot, Eestolano, CapitalR, Freeagent24, Unixguy, Boborok, Krauss, Thijs!bot, Nachmore, Ramckay, Aitherios, Lfstevens, Digitalfunda, Schily, Magnus Bakken, MartinBot, Aladdin Sane, Sidhekin, LokiClock, Philip Trueman, Maccam94, J3gum, AlleborgoBot, Kbrose, Vpg1986, Ccyber5, SieBot, Dwandelt, Coremayo, ClueBot, 1ForTheMoney, Chiisaisu, Boleyn, Bobbozzo, Dekart, Addbot, LPfi, Taktoa, Yobot, AnomieBOT, HughesJohn, Obersachsebot, Xqbot, Fumitol, Antarello, Tuplanolla, Arcwsimd, Czemstmyr, Lotje, Jdmartin86, EmausBot, WikitanvirBot, ZéroBot, John Cline, Can.already, ClueBot NG, Baseball Watcher, Marechal Ney, Sanju uthaiah, Shanecrosby, Lyle Stephan, Cyberbot II, ChrisGualtieri, Gmkwo, F6ERP.7vC7v, Dabreese00, Shadab.adel, Koenw and Anonymous: 157

10.2 Images

- **File:File_table_and_inode_table.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/f8/File_table_and_inode_table.svg *License:* CC BY-SA 4.0 *Contributors:* Own work *Original artist:* Qwertyus

10.3 Content license

- Creative Commons Attribution-Share Alike 3.0