Engineer

Computer file

a named set of data stored as one unit

What it is

A computer file is what a program asks the operating system for when it wants the same bytes back tomorrow: it opens the file by name, reads or writes, closes it, and trusts that the next program to open that name will see what was left there. Documents, photos, programs, settings and logs are all files, and on Unix-like systems so are several things that are stored nowhere at all.

POSIX, the standard behind Linux, macOS and the other Unix-like systems, defines a file as an object that can be written to, read from, or both, with attributes that include its access permissions and its type. It then lists the types: regular files, directories, symbolic links, FIFOs, sockets, and character and block special files, which stand for devices such as a terminal or a disk. When people say file they nearly always mean the first kind, which the standard describes as a randomly accessible sequence of bytes on which the system imposes no structure of its own. The system neither knows nor cares whether those bytes are a spreadsheet, a photo or a page of a database; structure belongs to the program that reads them. POSIX even declines to separate text files from binary ones, while noting that many utilities only give meaningful output on text.

The name is held outside the file. A directory is a list of entries, and each entry links a filename to a file. A filename is a single component that may contain any byte except a slash and the null byte, up to a limit the system sets; a pathname such as `/srv/site/index.html` is the chain of names the system walks to reach it. Because names live in directories, one file can have several of them, and the system keeps a count. When the last name is removed and no process still has the file open, the space is freed.

Everything else the system knows about a file is metadata, and a long listing shows most of it:

```
$ ls -li notes.txt
1048577 -rw-r----- 2 ana staff 4096 Oct 16 09:14 notes.txt
```

The first number is the file serial number, unique within its file system: that number is the file, and the names only point at it. Then come the permission bits for the owner, the group and everyone else, the number of names the file has, the owner and group, the size in bytes, and the time of last data modification. POSIX keeps two more times beside that one: the last data access and the last change to the file's status. A program that has opened a file holds a file descriptor, a small integer private to the process. Descriptors 0, 1 and 2 are standard input, output and error, which is why a shell can send a program's output into a file without the program knowing.

Use the concept in a decision

Put "What is a computer file?" to work in a real decision. We can help relate the concept to the system you are building or buying. Day-to-day care of servers, mail, accounts and updates, so the things everybody relies on stay boring.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Ran 10 PostgreSQL and MS SQL servers on Ubuntu — 10 database servers deployed and kept healthy

Ran 10 PostgreSQL and MS SQL servers on Ubuntu Get in touch →

Why it matters

Files are the oldest interface between programs that were never introduced to each other. A tool written decades ago can hand data to one written yesterday through a file, with no agreement beyond the bytes and the name. Configuration, logs, exports, backups and the content of most websites all travel this way, so keeping files trustworthy is most of keeping a system trustworthy.

The metadata is where the subtle trouble lives. The modification time says when the bytes were last written, which is a different fact from when what they say last changed, and the two part company the moment a tool rewrites a file with identical content. Ownership is stored as a number, so the same file can belong to a different person on another machine.

Then there is durability. A write that returns has handed the bytes to the operating system, which may keep them in a cache. POSIX says cached data reaches storage when a program asks for it with a call such as `fsync`, and may

get there earlier on its own, and it warns that after a crash later operations may be on disk while earlier ones are missing. It gives the example of a rename that leaves the new name created and the old one still in place.

Finally, a file is one copy on one device until someone arranges otherwise. Storage fails in ways that depend on the medium, and the only protection that holds across all of them is a second copy somewhere else that has been read back.

What it looks like in practice

A file's modification time records when its bytes were last written, which is a different moment from when its content last changed, and the site's sitemap once showed the gap: its dates put 172 of the 176 pages at one identical moment of change. The fix traced the date back to the file itself: the content export writes every file afresh on each run, so whatever a page said, one sync gave the whole site a single timestamp. The date now comes from version history through a fallback chain that tries an explicit field, then the commit history, then the file. The page records the trap in that chain: the literal field name has to be in the list, or the generator never reads it. A missing date leaves the field empty rather than holding a guess, and, at the recorded implementation stage, a re-sync after a month of edits touched eight files of 186.

The company's restore-test implementation ends with a file being read back. Its systemd units support a database dump where configured, an encrypted, deduplicated restic snapshot sent over SFTP to storage at another provider, retention pruning and an integrity check. A separate monthly workflow restores a known file to a scratch location and compares its bytes, so the intended test is a restore rather than an upload. The implementation record dated 2026-10-27 says the timers were disabled without a configured repository and records no completed live restore. Even the passphrase is a file, on purpose: the units read it from a protected file rather than a systemd-parsed environment value, whose escape handling could turn a backslash into a different passphrase from the one used to create the repository.

Years earlier the job was rescuing files, often the only copy, from failed hardware: data recovery across SD cards, spinning hard drives, SSDs, RAID arrays, external drives and Mac systems. Each of those fails and gives up its data

differently, and a Mac filesystem has its own quirks, so the work was as much about the right approach for the medium as about any single technique. The measure was the plain one: files handed back intact.

Where it goes wrong

Every page looks changed at once. A copy, checkout, restore or export rewrites the file and resets its modification time, and every system downstream that trusts that time sees a change that never happened. A copy that preserves times does the opposite and hides a change that did. Take content dates from the content or from version history, and leave the file time to mean what it means.

A reader finds half a file. A program that rewrites a file in place leaves a window in which another process, or the next boot, sees part of the old version and part of the new. Write a temporary file in the same directory, call `fsync` on it, then rename it over the original. POSIX requires that the replaced name stays visible throughout a rename and points at either the old file or the new one.

A script trips over a legal name. A filename may contain spaces, newlines and a leading dash, and a script that splits names on whitespace or passes them to a command unquoted will act on the wrong files. POSIX's own advice is to avoid a leading hyphen and to stick to letters, digits, the full stop, the underscore and the hyphen for names that must travel. Quote every expansion, and end option parsing with `--` before a list of names.

A secret file is readable by everyone. The permission bits on a new file come from the defaults of the process that created it, and those defaults usually let the group and others read. A key or passphrase written that way is shared with every account on the host. Create secret files with owner-only permissions from the start rather than tightening them afterwards.

Further reading

- NIST CSRC glossary: file — the definitions used in NIST security publications, each with the document it comes from.
- POSIX.1-2024 Base Definitions: Definitions — the formal meaning of file, regular file, filename, pathname, directory entry and file descriptor.
- POSIX.1-2024 Base Definitions: General Concepts — permissions, the three file times, filename portability and what a crash can leave behind in the cache.

- POSIX.1-2024: rename() — the guarantee that a name being replaced points at either the old file or the new one throughout, which safe saving relies on.

Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is a computer file?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/file/