Engineer

Operating system

the software that manages hardware for every other program

What it is

An operating system is the software that starts first on a computer and stays running underneath everything else, sharing the processor, memory, storage and network out among programs and giving all of them the same ways to read files, use devices and reach the network. It is usually shortened to OS. Windows, macOS, Android, iOS and the Linux distributions such as Debian and Ubuntu are operating systems.

NIST's glossary carries two descriptions that fit together. One calls it the master control application that runs the computer: "the first program loaded when the computer is turned on", whose main component, the kernel, stays in memory at all times, and which "sets the standards for all application programs", such as the web server. The other, from NIST's guide to operational technology security, calls it "an integrated collection of service routines for supervising the sequencing of programs", which may perform input and output control, resource scheduling and data management, and which gives application programs the fundamental commands for controlling the computer.

The kernel is the core, but an operating system is more than its kernel. Around it sit the system libraries that programs call, the shell and the everyday command-line tools, a supervisor that starts services at boot and restarts them, a package manager that installs and updates software, and a large body of default configuration that decides what is switched on, where logs go and who may log in. A Linux distribution assembles all of that around the Linux kernel, which is why two systems with the same kernel can behave very differently.

Most of what an operating system offers a program is a short list of abstractions that stand in for the hardware. POSIX, the standard behind Unix-like systems, defines them precisely. A process is an address space in which one or more

threads run, together with the system resources those threads need. A file is an object that can be written to, read from or both, with access permissions and a type: regular file, directory, symbolic link, FIFO, socket, or a character or block special file standing for a device. A block special file gives access to a device in a way that keeps its hardware characteristics out of view. A device itself is defined as “a computer peripheral or an object that appears to the application as such”. POSIX’s stated purpose is to make applications portable at the source code level, so a program written to these interfaces can be compiled on another conforming system and run.

Memory gets the same treatment. The Linux kernel’s documentation starts from the fact that physical memory is a limited resource and describes virtual memory as the layer that abstracts its details from application software. Each program sees an address space of its own, and the operating system decides, moment by moment, which parts of which programs are held in real memory and which are compressed, written out to disk or dropped to be read back later.

Use the concept in a decision

Put “What is an operating system?” 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

Choosing an operating system settles a great deal at once: which software will run, which tools the people running it already know, who supplies security fixes and for how long, and what a support contract, if any, will cost. Changing it later means moving every program and every habit built on it, so the choice usually outlives the hardware it was first made on.

It is also the main boundary of trust inside one machine. Users, groups, file permissions and the separation between processes are all operating system features, and they decide how far a flaw in one program can reach. A machine on which every service runs with full rights has one wall left, the network edge.

Its defaults are decisions somebody else made. A fresh install comes with services enabled, files in agreed places and settings already chosen. Most are sensible; a few will not suit the work, and nobody is told which. Running a machine well includes knowing which defaults you have accepted.

And it decides how far code travels. A program that keeps to the portable interfaces moves to another system with a rebuild. A program or script that leans on one system's extensions moves only as far as those extensions do.

What it looks like in practice

On a laptop, a macOS application's diagnostic reports recorded 111 GB of compressed pages on a 36 GB machine on 2026-09-01. A separate footprint sample recorded growth of about 41 MB per second during a connection retry loop emitting core events; those are separate observations, not one continuous measurement. The event path was changed to bound buffers, filter subscriptions by event type and give logging a rate budget. A full test-run comparison recorded core log lines falling from 610,996 to 1,411. Tests defend the bounded design, but that comparison is neither a production benchmark nor a measured post-fix memory figure. Separate shutdown timeouts required their own correction.

On the company's host, an automation role had written its SSH settings on every run for months when an access report read the daemon's own resolved settings and found two of them disagreeing. The account of hardening SSH with three-stage validation gives the cause: the operating system ships its own defaults uncommented, above where a drop-in file lands, and for those two settings the first occurrence wins. The fix was a filename prefix that sorts ahead of the vendor's, and 24 directives are now asserted against what the daemon reports back.

The same host met another of the system's choices when its intrusion-banning service was set up. The log source is set to detect because the operating system ships no traditional authentication log, and the wrong choice fails silently by watching a file that never appears.

Where it goes wrong

A script carries the habits of the system it was written on. A shell script written on a Mac and run on a Linux server, or the other way round, meets the same command names with different options. The POSIX `sed` accepts `-E` and `-n`; editing a file in place with `-i` is an extension, and the two common versions take it differently:

```
sed -i 's/old/new/' notes.txt     # GNU sed, on most Linux systems
sed -i '' 's/old/new/' notes.txt  # BSD sed, on macOS
```

Each form breaks on the other system. Keep to what POSIX defines where it is enough, run the script on the system that will run it in production, and install the same tool on both when an extension is worth having.

Two machines with the same operating system differ. “Debian” names a family of releases, and two hosts both described as running it can be years apart in release, kernel and package versions, with different packages installed and different local changes. Read what each machine actually is before comparing them:

```
cat /etc/os-release
uname -srm
```

The first prints the distribution’s name and release, the second the kernel’s name and release and the machine’s processor type. Build every host from the same written description and keep the release in the inventory.

Services run as the administrator. A service running as root on a Unix-like system can read every file and change every setting, so any flaw in it is a flaw in the whole machine. Give each service its own unprivileged user, with write access only where it writes, and grant extra rights one at a time when the service shows it needs them.

A release outlives its security support. Every release stops receiving fixes on a published schedule. Debian’s security team, for example, supports each stable distribution for three years from the day it is released. A host left on an old

release keeps working, which is exactly why it gets forgotten. Record each host's end-of-support date beside its other renewal dates and plan the upgrade well before it.

Further reading

- operating system — NIST CSRC Glossary — both definitions, from the master control application to the collection of service routines.
- POSIX.1-2024, Base Definitions: Definitions — the exact wording for process, file, device and the other words an operating system is built from.
- POSIX.1-2024: sed — the options every conforming `sed` accepts, as a measure of what a portable script may use.
- Linux kernel documentation: Concepts overview — virtual memory, the page cache and reclaim, as the kernel itself describes them.
- Debian security FAQ — how Debian handles security fixes and how long a stable release receives them.



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 an operating system?” 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/operating-system/