THE LINUX FOUNDATION | Research

Linux 35

35 Years of Linux

The Open Source Engine of Global Infrastructure and Innovation

Anna Hermansen, The Linux Foundation
Hilary Carter, The Linux Foundation
Daniel German, University of Victoria

Foreword by Dirk Hohndel,
Ericsson Software Technology

October 2026

35 Years of Linux

Today, Linux powers 4B Android devices, 30M Chromebooks, 200M+ servers, all of the top 500 supercomputers, and Apple's 5G modems.

Linux runs automobiles, global stock exchanges, household electronics, industrial equipment like elevators and conveyor belts, and satellites and space infrastructure.

The Linux kernel sees nearly ten changes every hour, and a new release taking place every eight to nine weeks.

The kernel continues to attract new developers, with the 7.0-7.2 releases breaking records for first-time contributions.

The first two quarters of 2026 saw over 4,200 contributors each quarter, ensuring continuous improvement and an active base.

In mid-2026, 131 contributors accounted for 51% of contributions and 7 organizations for 51%, translating to high contributor diversity and a resilient development process.

Corporate engineers today account for >80% of all Linux contributions, with organizations like Google, Ericsson, Fujitsu, Intel, Meta, Huawei, Samsung, and SUSE working collaboratively.

Over 50% of contributors in 2026 are contributing consistently over quarters, indicating a stable and engaged community.

The development of Kernel source code evolved in three stages: pre-version control, BitKeeper, and Git. Each transition encouraged greater complexity and maturity for the project.

Linux revolutionized software engineering by pioneering review-driven collaboration and creating Git, the transparent, distributed workflow of modern development.

Linux powers virtually all major visual effects render ranches and enables cross-architecture PC gaming through Valve's Proton compatibility layer.

Global stock exchanges and banks rely on Linux for low-latency trading, risk modeling, and shared open source supply chain resiliency.

Copyright © 2026 **The Linux Foundation** | October 2026. This report is licensed under the **Creative Commons Attribution-NoDerivatives 4.0 International Public License**.
Please cite this research report when using the infographics. See guidance on the report's back page.

35 Years of Linux

Linux forced open Wi-Fi compliance standards across chipmakers, ultimately coming to power global cellular modems, consumer routers, and enterprise access points.

In healthcare, embedded Linux powers medical equipment, diagnostic tools, and genomic research while adhering to international safety certification standards.

Autonomous space missions, satellite constellations, and commercial flight drones use Linux to execute real-time navigation algorithms and complex telemetry processing.

Automotive Grade Linux unifies the vehicle software stack under a collaborative framework with long-term support for safety-critical software.

Linux's stability, customizability, and scalability disrupted proprietary computing markets, and it now powers the majority of websites, mainframes, servers, and mobile devices.

Cloud native innovation was made possible due to core primitives engineered directly into the Linux kernel, which is now the essential runtime of modern software workloads.

Thanks to real-time kernel capabilities and hardware portability, Linux dominates local edge devices, regional data centers, and 5G network towers.

Linux has evolved to power compute-intensive parallelized AI workloads, while AI tooling has simultaneously transformed kernel maintenance, patch reviews, and bug detection.

Floods of AI-generated patches to the Linux kernel's private security list has prompted documentation updates to clarify vulnerability reporting standards and restrict AI bot submissions.

To eliminate major vulnerability classes, the Linux community launched KSPP to integrate proactive structural defenses and adopted Rust for memory-safe driver development.

The LF Mentorship program addresses entry barriers through stipends, webinars, and mentorships to foster open source diversity, technical skills, and the "people" aspects of software engineering.

By enabling accessible digital payments, agricultural tools, clean energy, and global connectivity, the Linux kernel silently powers essential daily experiences for millions worldwide.

Copyright © 2026 **The Linux Foundation** | October 2026. This report is licensed under the **Creative Commons Attribution-NoDerivatives 4.0 International Public License**.
Please cite this research report when using the infographics. See guidance on the report's back page.

Contents

Foreword **5**
Remembering those we have lost **7**
Introduction **8**
The Linux architecture **9**
Three eras of Linux kernel development 10
Linux's impact on software engineering practice 12
Linux's industry impact **13**
Financial services 14
Communications and networking 15
Automotive 16
Motion pictures and gaming 17
Healthcare 18
Industrial manufacturing and civil infrastructure 20
Space, drones, and autonomous systems 21
Linux's technological impact **24**
Personal and enterprise computing 24
Cloud computing and containers 27
Edge 29
Artificial intelligence 30
Sustainability and the future of the kernel **33**
Security 33
Mentorship for the next generation 34
The kernel's human impact **37**
Conclusion **38**
Acknowledgments **39**
About the authors **40**
References **41**

Foreword

When people talk about the history of a project, they start with the present and look back, invariably putting what happened "back then" in the context of "what we know today". But when Linux started, the trajectory of the project was of course unknown, and not even in their wildest dreams would anyone have predicted the future as it unfolded. Initially, Linux was just Linus Torvalds's silly experiment and an attempt to learn 386 assembler and understand the memory management functionality of that processor. It was supposed to be a small hobby project, something that just a few people would ever see and even fewer would ever care about.

Linus has always been quick with a bold quote and has even more often been misquoted and had his statements about his goals misinterpreted (famously: "World Domination, Fast"). In reality, even in 1994, when he released version 1.0 of the Linux kernel three years after starting the project, he was still thinking of Linux as his small hobby project that just happened to have attracted a lot more people than he expected.

That mindset had a lot to do with the initial success. What made the success sustainable was and is Linus's ability to morph this mindset over time, to learn from what works, and most importantly to be willing to make changes when things don't work. Sometimes very significant changes. For example, when he decided to change the release model of the kernel to a completely new model, which, frankly, at the time was widely perceived as impossible. A new release every 9-10 weeks? A relentless, predictable cadence that all work should follow? No release plans, no visions, no carefully scripted months-long processes? Most people assumed this approach would fail quickly. Today, more than 20 years later, that's still the process used and it is a major reason for Linux's success. But back in 2005 it was simply a reaction to the old process that had failed, and the willingness to make fundamental changes, learn from the failure, and move on.

And that same pattern was repeated over and over again; keep doing what works, question what doesn't, and be willing to make the changes to address the issues. You don't need to be able to predict the future. You just need to be able to figure out how to do better tomorrow than you did yesterday.

Or, to quote Linus: "I am not a visionary. I do not have a five-year plan. I'm an engineer. [...] I'm looking at the ground, and I want to fix the pothole that's right in front of me before I fall in. This is the kind of person I am."

When talking about 35 years of Linux, we really should be talking about the incredible perseverance (which sounds much better than stubbornness) that led to a series of small decisions which grew a small hobby project into one of the most important and influential software projects ever.

Dirk Hohndel
Head of Ericsson Software Technology

Linus Torvalds and Dirk Hohndel speaking at Open Source Summit North America, Minneapolis, May 2026.

Remembering those we have lost

For 35 years, the Linux kernel has grown through the shared efforts of people who brought their knowledge, time, and ideas to a common endeavor. Contributions exchanged across mailing lists became collaborations, collaborations became friendships, and successive generations of developers found colleagues ready to teach, encourage, and challenge them.

Some of the people who helped create this community are no longer with us.

Their work remains part of the kernel and continues to shape the systems built upon it. Yet their legacy extends beyond the software itself. The strength of the kernel community has always depended on human connections: on people willing to explain a difficult problem, help someone find their way, offer a fresh perspective, or bring humor and warmth to the work.

Those relationships have an enduring effect. Developers carry forward not only technical knowledge, but also lessons about how to collaborate, how to approach difficult problems, and how to help the next person succeed. In this way, the influence of those who came before continues through the people they taught, supported, and inspired.

Their absence reminds us that a community is made up of individuals, each with a distinctive voice and presence. We miss the conversations they would have joined, the questions they would have answered, and the perspectives they would have brought. We miss the people behind the commits—their curiosity, generosity, humor, determination, and friendship.

As we celebrate 35 years of Linux, we pause to remember members of our community who have passed away. We honor their contributions to the kernel and to open source, while also recognizing what they gave to the community itself: knowledge shared, friendships formed, traditions established, and countless small acts of support that helped others continue the work.

The kernel will keep evolving, carried forward by new generations. The work of those who came before remains woven into that continuing story, as does their influence on the people who knew them.

We remember them with gratitude and affection. We are grateful for what they gave to Linux and to one another, and we feel their absence still.

Introduction

Thirty five years ago from Helsinki University, Linus Torvalds released a technology to the world that would irrefutably change the world. That technology was the Linux kernel, and at the time, no one, not even Torvalds its creator, could have imagined that an operating system kernel hobby project would grow to underpin the world's digital infrastructure. Today, the kernel can be found powering entire industries, spawning new companies, enabling new products, and inspiring one of the largest software collaborations in history.

"Why is the Linux kernel so important as a piece of software?" asked Michael Dolan, senior vice president of legal and strategic programs at the Linux Foundation. "Because it's openly available to anyone at any time to instantaneously build things on a proven operating system most of the world's software supports."[1] The open license of a technology as powerful as the Linux kernel was, and remains, a force multiplier for its proliferation. That openness, in combination with Git, another Torvalds creation facilitating version control and mass collaboration on code, has enabled thousands of developers to work on the project across more than 350 companies. It has an important economic impact as well, providing the means to develop low-cost, affordable software solutions. For example, due to the affordability of Android devices, the market share of Android in India is upwards of 95%.[2] This accessibility means a larger share of Indians are connected online.

Since its first release in 1991, the Linux kernel has grown to see nearly ten changes happening every hour, contributed by a vast developer community, with a new release taking place predictably every eight to nine weeks. Developers have always been the source of Linux's astounding innovation path. After 35 years, one might assume the kernel developer community has dwindled based on peer projects, however, the reality shows otherwise. As Jonathan Corbet has noted, the 7.1 kernel release in 2026 included work from 530 first-time contributors. That was a record, but a short-lived one. As of the 7.2 kernel candidate release, the kernel has merged patches from 613 first-time contributors.[3,4]

Today, Linux powers four billion Android devices, 30 million Chromebooks annually over the past decade, Apple's 5G modems, 200 million-plus servers, all of the top 500 supercomputers, automobiles, global stock exchanges, household electronics like TVs, thermostats and washing machines, industrial equipment like elevators, conveyor belts and wind turbines, as well as satellites and space infrastructure. This list by no means is exhaustive. To quote Linux kernel maintainer Greg Kroah-Hartman, many of these figures represent "a rounding error." In short, he says, "it all runs on Linux."[5] From the kernel, other projects have spun off and become critical components of our software stacks, such as Docker, eBPF, and device trees. These byproducts in turn have positive implications on the rate of change of the kernel itself, but have also impacted the future paths of alternative operating systems.

This report is a celebration of the Linux kernel project on the occasion of its 35th anniversary. It will explore the project's significant development milestones and architecture, with a focus on both the project's present day impact within industries and across technologies, and the contributor community dynamics that have shaped and will continue to shape the Linux kernel long into its future.

The Linux architecture

Linux grew out of a need to fill a gap in the open source computing system. In 1983, in response to Unix's growing commercialization and restrictions, Richard Stallman launched GNU to build a completely free, Unix-compatible system and developed the GNU General Public License (GPL) to protect its freedoms.[6] By the early 1990s, GNU had much of the needed stack, but not a production-ready kernel. Linus Torvalds's Linux kernel, which started using the GPL-2.0 license in 1992, became the missing piece that made a completely free Unix-like system viable.

Torvalds created the Linux kernel and posted it publicly with the hope of receiving developer feedback on his project. According to one interview, he did not consider this an open source act, but it became associated with the movement as contributors continued to add features.[7] Alongside the kernel, the growing use of software licensed for collaborative use and development led to the coining of the term "open source."[8] Contributions to the kernel grew dramatically from 1991 and started to stabilize between 12-15,000 commits per release in 2014.[9] As of writing this report, Linus had released 7.2, which saw 16,418 non-merge commits from 2,652 developers—a new record for the kernel.3 The open and collaborative development process structured around continuous release cycles has allowed the kernel to evolve and mature organically and quickly over time. Each release incorporates various features that keep the kernel relevant, such as version 2.0's symmetric multiprocessing support so that Linux could run on systems with multiple CPUs, version 2.6's advanced hardware support and improved scalability, version 4.0's live kernel patching, and version 6.12's mainlining of real-time support.[10,11]

The broader Linux architecture is made up of different components, which includes the following:

- The hardware layer, such as the CPU, RAM, and input/output devices
- The Linux kernel, which manages processes, memory, files, and devices, and acts as a communication line between the hardware and applications
- Libraries, that provide functions for applications to use to communicate with the kernel
- The shell, or command line
- The user space, where applications are run.[12]

Linus's decision early on to port Linux—which initially ran on only 32-bit Intel x86 machines—to the 64-bit, RISC-based DEC Alpha machine forced a rewriting to make the kernel portable, which set the stage for Linux to run on everything from watches to supercomputers.[13] Since this rewrite, the architecture allows anybody to configure the kernel in many different ways, specifically across different hardware environments. There are also hundreds of Linux distributions, all developed for different uses of the kernel. Major ones include Debian, which became the basis for Ubuntu and Raspberry PI OS; Red Hat's enterprise distro, which produced Fedora; and Android. This modularity has led to significant proliferation of the kernel today, with Linux powering 200 million servers, billions of Android systems, cloud platforms, supercomputers, embedded systems, Chromebooks, home appliances, cars, enterprise infrastructure, and spacecraft.

Linux Kernel Developer Series: Jonathan Corbet

Jonathan Corbet's journey into kernel programming began in the early 1980s during his undergraduate studies, where system administration expert Evi Nemeth gave him access to the source for an early BSD release on a VAX system and tasked him with fixing memory management so large jobs wouldn't starve other processes. After his studies, he worked with early meteorological "free" software at the National Center for Atmospheric Research (NCAR) and wrote and distributed code on DECUS tapes. The release of the Linux kernel attracted his attention quite quickly, where he first added tweaks to the X window system for his work at NCAR. Eventually, he left NCAR to work on Linux full-time.

His career as a kernel maintainer evolved from there, and he took on roles and projects that placed him in the pulse of the community. Before the Git era, Jonathan's first contributions meant emailing Linus Torvalds directly and inspecting released patch sets to see if a contribution had been accepted. He co-founded LWN.net in 1998, which became the go-to independent news publication covering kernel development and community dynamics. He co-authored the book Linux Device Drivers and later assumed maintainership of the kernel's official documentation. He also served on the Technical Advisory Board in its very first election and subsequently for almost two decades, helping bridge communication between disparate subsystems, corporate entities, and developers. He said about his board role, "it gave me a much better view of what's going on in our community... trying to connect people who I think should be talking to each other and who aren't, who are working on similar problems or can somehow help each other."

About why he continues to work in the community, he expressed, "we all launched into this with the idea that we could change the world in various ways. It was initially a fight against the takeover of proprietary software and systems that we couldn't control... and we succeeded beyond our wildest dreams in terms of changing the world. I'm not convinced the world we have created is necessarily the one we had in mind, but that's the nature of changing the world."

Listen to Jonathan's full interview, **Life as a Linux Kernel Developer and Maintainer with Jonathan Corbet**[14]

A special thanks to Jonathan and the LWN.net team who have been tirelessly documenting the evolution of the Linux kernel and the community for 28 years. The open source community, including the authors of this report, has learned from and leaned heavily on the thorough and meticulous work of LWN.net contributors.

Three eras of Linux kernel development

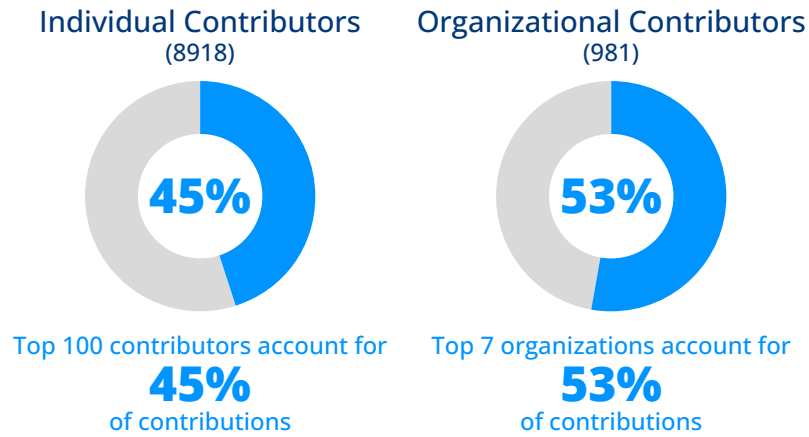
The cregit tool, developed by Daniel German, provides a granular view into Linux kernel development, tracking the evolution of the kernel source code in its three stages: pre-version control, BitKeeper, and Git.[15] During the pre-version control era, which lasted from September 1991 until February 4, 2002, only Linux releases themselves provided a window into its development scale. By the early 2000s, Linux had outgrown the patch-centered workflow that had served it during its first decade. The kernel had become a large, fast-moving project with many subsystems, maintainers, architectures, drivers, and commercial stakeholders. On February 4, 2002, the distributed version-control system BitKeeper was adopted to give maintainers a better way to manage independent streams of work, preserve history, merge changes, and coordinate development at scale, but encouraging developers to use it, in Torvalds's words, "... the real payoff comes if main developers start using bk too [...]. That will take some time."[16] This second era of development marked the start of the Linux kernel's commit history.

This transition accelerated Linux development. By the end of 2003, the community released a historic kernel: Linux 2.6. This series is now considered a landmark branch for the kernel, improved scalability, responsiveness, threading, I/O, CPU and hardware support, and suitability for enterprise and server workloads. Linux was becoming a foundation for modern computing infrastructure.

The process matured alongside the system. BitKeeper reinforced a more distributed, maintainer-centered model of collaboration, while practices such as the Signed-off-by line and the creation and adoption of the Developer Certificate of Origin strengthened accountability and provenance.[17] Linux development evolved into a structured process for coordinating distributed work while preserving review, responsibility, and traceable change. During the BitKeeper days the kernel grew from 8,740 files to 17,844 files.

The current version control system, the Git era, began on April 16, 2005. When the Linux community lost the free license to use BitKeeper, Linus Torvalds was prompted to develop a new tool for version control.[18] He created the tool with a few principles in mind, ensuring it was fast, simple, supported, distributed development, and able to handle large projects.[18] He also open sourced the project. Git has transformed not only the kernel's development process, but virtually all open source projects, and has certainly played a critical role in the kernel's highly distributed, diverse, and healthy contributor base (see **FIGURE 1**). The git tool's impact on software engineering is discussed in the following box.

FIGURE 1: Kernel contributor statistics, September 2025 to August 2026



Average Contributor Retention Rate
63%

From September 2025 – August 2026, the kernel had 8918 active individual contributors, with the top 100 accounting for 45% of all contributions. 981 organizations actively contributed, with 7 of those accounting for 53% of all contributions. Contributor retention was between 55%-75%.

Source: *The Linux Kernel Organization Insights, LFX. Available from:* **https://insights.linuxfoundation.org/project/korg?timeRange=past365days&start=2025-08-28&end=2026-08-28**[20]

Attendees of the Embedded Open Source Summit, Seattle, April 2024.

Linux's impact on software engineering practice

Linux's influence goes beyond its use as an operating system. As a major catalyst for modern collaboration practices, it changed software engineering. Linux has consistently pushed forward the idea that development should be incremental, review-driven, and version-controlled. The growing scale and geographic distribution of kernel development led Linus Torvalds to rethink how a large, distributed group collaborated to develop software and led him to create Git. Git perfected distributed version control around the kernel's workflow: changes are proposed, reviewed, discussed, tested, and integrated through a visible process.

Git, in turn, carried the practices of kernel development beyond the kernel itself. Designed to support easy, fast, and distributed work at the scale of Linux, it was soon adopted by other open source projects and later by commercial software teams. As hosting platforms, code review systems, and continuous integration tools grew around Git, practices first sharpened in the kernel community became part of the everyday infrastructure of software development.

> **Linux's legacy, therefore, is not only the system that runs much of today's infrastructure. Through Git, it also helped define the development model behind much of today's software: distributed, incremental, review-driven, and built around transparent histories of change.**[19]

Linux's industry impact

Academic and economic impact analyses demonstrate that if open source software like Linux were suddenly eliminated, global GDP for the average nation would be over 2% lower in the long run due to lost operational efficiencies, increased total cost of ownership (TCO), and duplicated research costs.[21] Today, corporate engineers account for more than 80% of all kernel contributions, with thousands of unique organizations having at least one active contributor since the project's inception.[20] Global enterprises like Fujitsu, Huawei, and Samsung from Asia, Ericsson, Igalia, and SUSE from Europe, and US tech giants like Intel, Red Hat, Google, Meta, AMD, NVIDIA, and Qualcomm, work collaboratively in the project, alongside countless others. For a list of the top contributing organizations in 2026, see **FIGURE 2**.

Notably, Linux has become the universal foundation for software-defined industries.[22] In this section, we describe the role of Linux in shaping industry-specific innovation, and the present day, widespread collaboration on common industry challenges and pre-competitive software baselines.

> **Linux has become the universal foundation for software-defined industries.**

FIGURE 2: Top organizations contributing to the Linux kernel, September 2025 – August 2026

	Organization	Total Contributions
1	Linux Kernel	48,192
2	Intel Corporation	32,276
3	Red Hat	17,935
4	Advanced Micro Devices, Inc.	15,491
5	Google	13,786
6	SUSE	12,027
7	Meta Platforms, Inc.	7,422
8	NVIDIA	7,045
9	Linaro	6,912
10	Qualcomm Inc	6,044

Organizations are ranked by the number of contribution activities performed by contributors on their behalf.

Source: *The Linux Kernel Organization Insights, LFX. Available from:* **https://insights.linuxfoundation.org/project/korg?timeRange=past365days&start=2025-08-28&end=2026-08-28**[20]

Financial services

When the Linux kernel emerged in 1991, financial services likely would have been among the last industries expected to embrace and deploy it. Banks and exchanges ran on proprietary Unix variants and mainframes. The idea of mission-critical trading infrastructure resting on freely available, community-maintained code would appear at odds with an industry defined by risk aversion and regulatory scrutiny. Yet by the late 1990s, the rationale for Linux spread like wildfire, and the financial services industry soon became "one of the first mainstream corporate markets to adopt Linux in mission-critical applications."[23]

The kernel's transparency, rapid patching cadence, and absence of per-system licensing fees made it attractive to an industry needing to scale electronic trading and back-office processing without the vendor lock-in of proprietary Unix. Early adopters initially used Linux first for peripheral workloads such as web servers, batch processing, and test environments before moving it into the data center core.

The use cases within the sector multiplied quickly. Exchanges and market-makers adopted Linux for its low latency and tunability at the network and scheduling level, making it particularly effective for the pace needed for clearing and trading.[24] Investment banks built grid-computing environments for risk calculations and derivatives pricing atop Linux clusters, finding it cheaper and more flexible than commercial Unix at scale. Eventually, Linux would expand to underpin the world's largest exchanges themselves: New York, Chicago, London, Frankfurt, Shanghai, and others migrated core trading platforms to Linux-based systems, citing performance and cost advantages over legacy infrastructure.[25]

The impact went beyond cost savings. Linux adoption normalized the idea that open, collaboratively maintained software could meet the reliability and auditability demands of a heavily regulated sector. It demonstrated that transparency — the ability to inspect, patch, and verify code — was compatible with, and could even strengthen, the risk controls that regulators required. This shift in mindset mattered as much as the technology itself: it made the case that open source could be trusted with systemically important infrastructure.

> **Linux adoption normalized the idea that open, collaboratively maintained software could meet the reliability and auditability demands of a heavily regulated sector.**

That precedent set the stage for formal industry collaboration. Rather than each institution independently rediscovering the same patterns for adapting open source to regulated environments, the industry moved toward pooling that work. This logic led to the founding of the Symphony Software Foundation, which then became **FINOS** (the Fintech Open Source Foundation), today part of the Linux Foundation. The community now unites more than 100 member organizations—banks, fintechs, and technology vendors—to build shared open source tools, standards, and governance practices for finance. FINOS Executive Director, Gabriele Columbro, reflects on the significance of Linux as a catalyzing force for the industry. He says, "Thirty-five years ago, few would have bet financial services' most critical infrastructure could run on a freely available kernel. Today, that same openness underpins our exchanges, our risk systems, and our industry's ability to collaborate on shared resilience."

Gabriele Columbro speaking at the Open Source in Finance Forum, London, June 2026.

Communications and networking

The advent of consumer wifi at the turn of the 21st century meant Linux-powered devices had to adapt. With the IEEE 802.11 standard and Apple's Airport technology, Linux developer Jean Tourrihes of HP was compelled to create Wireless Extensions (wext), a set of commands that allow applications to talk to wireless drivers.[26,27] As Wi-Fi scaled and improved, the hardware companies fought to keep their wireless logic proprietary, and the industry fragmented, with chipmakers having to constantly rewrite Wi-Fi logic from scratch for their individual drivers. Open source developers eventually embedded a universal, open source Wi-Fi compliance framework directly into the core kernel.[28] With this framework, if a chipmaker wanted their new chip to be bought by laptop manufacturers or cloud vendors, it had to work with the Linux kernel's open source architecture. As a result, Linux forced the entire semiconductor industry to cooperate on a single, shared software standard, radically accelerating the time-to-market for new Wi-Fi generations, driving down the manufacturing cost of wireless chips, and eventually accelerating the adoption of network functions virtualization, software-defined networking, cloud native networking, and hyperscale data center architectures.[28]

Linux went on to conquer the global Wi-Fi infrastructure market thanks to open source router and standards development. In 2002, to save on development costs, Linksys released the WRT54G router using a modified Linux kernel along with its source code.[29] GPL-2.0 compliance enforcement by the Software Freedom Law Center established that the license for the kernel could be enforced in court, but also unlocked the Linux-based router for community improvement and modification. This led to the development of now-ubiquitous projects like OpenWrt and a rush of cheap consumer routers that matched performance of expensive enterprise boxes.[29] Today, Linux is the underlying operating system running a home Wi-Fi router, enterprise access points, and smart home IoT hubs, and connecting to Linux-based TV boxes.[30,31,32,33] For example, all Samsung TVs run on Tizen OS, an open source, Linux-based software stack.[34]

The shift to commodity hardware created the foundation for a new generation of open source networking projects that build on the Linux kernel. Projects such as DPDK optimize packet processing performance, SONiC and DENT extend Linux into open network operating systems for switches, OPI standardizes programmable infrastructure for DPUs and IPUs, and LF Networking integrates orchestration, automation, and interoperability across telecom and cloud networks. Together, these projects represent the natural evolution of Linux into a common platform for programmable and open networking infrastructure.

As Arpit Joshipura, general manager of networking, edge, & IoT at the Linux Foundation reflects, "Linux transformed networking by making it programmable, open, and accessible to everyone. Today's networking projects build on that foundation to extend the same principles beyond the operating system, creating interoperable platforms for cloud, edge, telecom, and AI infrastructure. Open collaboration continues to be the catalyst that turns innovation into industry-wide adoption."

Automotive

The integration of Linux in the automotive industry has redefined the modern vehicle and created immense value for manufacturers and users, transforming vehicles from mechanical machines into sophisticated mobile data centers. This evolution is spearheaded by Automotive Grade Linux (AGL), a collaborative open source project hosted by the Linux Foundation that has replaced fragmented, proprietary software systems. While the Linux kernel provides the core engine for this transformation, the Yocto Project enabled car manufacturers to make a Linux distribution for their specific needs. The result was AGL's unified code base (UCB), a robust foundation that represents approximately eighty percent of the software stack required for a vehicle. AGL also addresses the automotive industry's need for long-term support, ensuring that vehicular software remains secure and maintainable over a fifteen-year vehicle lifecycle.

The real-world impact of this unified Linux distribution was famously validated with the launch of the 2018 Toyota Camry, the first major production vehicle to adopt the AGL platform for its infotainment system. This milestone signaled a shift away from isolated research and development, paving the way

Linux Kernel Developer Series: Paul E. McKenney

Paul E. McKenney has been a Linux developer for over 20 years, but his involvement in computing began before Linux, where he worked on DYNIX/ptx—a proprietary parallel Unix system for high-end database servers—and before that in network research. Paul first heard of Linux in the mid-1990s when asked to help retrieve data from a Mac. Shortly after, networking maintainer Alexey Kuznetsov requested a 1990 paper Paul had published on stochastic fair queuing. Kuznetsov implemented the ideas in Linux and cited Paul in the kernel code comments, making Paul's very first contribution a non-code bibliography entry.

At that time, Paul was working on DYNIX/ptx at Sequent, and when IBM acquired Sequent around 2000, Paul shifted focus to Linux development. He worked on a number of things, but he does not remember his first patch: "some got in, some didn't—it was really hard." He became the lead maintainer for Read-Copy Update (RCU), a critical and complex concurrency mechanism used pervasively across the Linux kernel. Paul created a 50-year plan for RCU maintainership in 2010 to onboard co-maintainers and ensure the subsystem would survive beyond his own tenure. Part of this maintenance, he said, is "codification and teaching... All the [RCU] nomenclature and all the definitions and all that are things that we had to come up with to talk to other people about it. And I've gotten better over 30 years of explaining to people."

Beyond writing code, Paul authored the open source textbook Is Parallel Programming Hard, And, If So, What Can You Do About It?, he created testing suites like rcutorture, he works on the Linux Kernel Memory Model (LKMM) project, and he actively mentors developers globally. When describing why he continues to work with the community, he said, "One thing is responsibility. The Linux kernel community put a lot of trust in me when they accepted RCU."

Listen to Paul's full interview, **My Life as a Linux Kernel Developer and Maintainer with Paul E. McKenney**[35]

for a broad coalition of members including Mazda, Mercedes-Benz, Subaru, Suzuki, and Panasonic, to name a few. According to AGL's Senior Director of Community, Walt Miner, "Toyota's commitment to AGL was reinforced in 2026 when it rolled out a refreshed UI design, using their open source embedded Flutter engine and the latest AGL components in the redesigned RAV4."

By contributing to a common platform, manufacturers no longer need to independently solve basic technical challenges like connectivity or hardware abstraction. Instead, they can leverage a consistent environment where software can be developed once and deployed across multiple vehicle models. This shared investment accelerates time-to-market and allows companies to refocus their resources on creating unique, branded digital experiences for their customers.

Today, AGL is powering the move toward the software defined vehicle (SDV) through the SoDeV reference platform, a framework that runs the AGL Linux distribution within isolated virtual machines. This architecture allows a single high-performance computer to manage safety-critical functions alongside instrument cluster and entertainment applications on the same hardware without interference. This ensures that the digital dashboard remains reliable even if a consumer app encounters an error. Industry research indicates that adopting such open, Linux-based architectures can reduce vehicle research and development costs by as much as forty percent, while the Linux Foundation reports that over one hundred fifty organizations have now unified around this production-ready standard to drive the next era of mobility.[36,37]

Dan Cauchy speaking at the Automotive Linux Summit, Tokyo, December 2025.

Motion pictures and gaming

In the entertainment vertical, the Linux kernel serves as the fundamental engine powering modern digital pipelines across motion pictures and gaming, and the catalyst for current open source collaboration within the industry. During the creative boom of the 1990s, visual effects (VFX) and computer graphics (CG) animation studios relied heavily on proprietary, high-cost Unix-based IRIX workstations from Silicon Graphics (SGI) to push the boundaries of digital storytelling. When SGI shifted its strategic focus away from the specialized, rapid technology needs of Hollywood filmmakers, a historic industry-wide coalition of studios banded together in June 2000, signing a joint commitment to transition their pipelines to the open source Linux operating system.[38]

The early days of this migration were defined by trial, error, and immense engineering ingenuity. James Cameron's Titanic (1997) became the first major motion picture to render the vast majority of its digital effects—specifically its complex water scenes—using a massive render ranch running Red Hat Linux. Shortly after, DreamWorks Animation's Shrek (2001) became the first entirely computer-animated feature film to be built completely within a Linux environment, leveraging over 1,000 Linux servers to process its complex volume and hair mechanics.[38] This shift allowed studios to escape costly hardware locks, using commodity Intel processors to achieve what technologists called "disposable computing." By combining their market weight, the studios successfully pressured major commercial software vendors to port core creative applications, like Alias Wavefront's Maya, SideFX's Houdini, and Pixar's RenderMan, straight to Linux.

> **Today, Linux has completely conquered the Hollywood production landscape, with virtually 100% of major visual effects, animation, and digital rendering clusters running natively on Linux node environments.[39]**

Building on this industry-wide collaboration on Linux, the Academy of Motion Picture Arts and Sciences partnered with the Linux Foundation in August 2018 to establish the Academy Software Foundation (ASWF) bringing the collaborative approach to Linux to software used by motion picture studios.

This year, new opportunities were created for Linux to impact the industry. Building on its initial 2018 release of the Proton compatibility layer, Valve has pushed the envelope of hardware independence by introducing Proton 11.0. This release integrates Wine 11 and a brand-new, kernel-level sync framework called ntsync, which drastically reduces CPU overhead for multi-threaded Windows games running on Linux. Even more revolutionary is Proton's breakthrough integration of FEX-Emu to deliver ARM64 compatibility. This allows games originally built for x86 architecture to run natively on mobile ARM64 hardware running Linux, laying the groundwork for Valve's standalone VR headset, the Snapdragon-powered Steam Frame, and allowing open source Linux environments to run complex PC titles on energy-efficient handhelds.[40]

Rachel Rose speaking at Open Source Summit North America, June 2022.

Healthcare

Healthcare has long occupied a paradox in Linux's history. As an industry built on data-intensive, computation-heavy workloads, it has been one of the slowest, comparatively, to formally adopt open source at scale. Unlike other industries,

the Linux kernel's presence in healthcare has not been defined by use in IT departments but rather in embedded systems such as medical imaging equipment, laboratory devices, and diagnostic workstations. Given the nature of healthcare, where a software failure can mean loss of life, the kernel's adoption path has been comparatively slow vis-a-vis finance or telecom. At the same time, the rationale for Linux in healthcare, where a stable, widely scrutinized, and cost-effective foundation, backed by a global developer community capable of identifying and patching vulnerabilities faster than any single vendor could alone, would make it a logical choice for the industry.[41]

The incorporation of Linux within the healthcare vertical did not happen overnight. As medical instrumentation became increasingly interconnected, manufacturers required a platform that could handle workflows such as multitasking, graphics, and networking. Embedded Linux filled this gap, allowing medical device manufacturers in particular to eliminate proprietary engineering silos and accelerate their time-to-market for their respective hardware and devices.

The healthcare industry's need for computing infrastructure that enabled cutting edge innovation while meeting stringent safety requirements soon gave rise to the **Enabling Linux in Safety Applications (ELISA)** project. Launched by the Linux Foundation in February 2019, ELISA brings together members to build shared tools and processes that help companies meet the safety requirements needed for certification. The project hosts a Medical Devices Working Group, which focuses specifically on IEC 62304, the international standard governing the medical device software lifecycle, which the FDA recognizes as a consensus standard supporting premarket submissions.[42]

Beyond the device layer, Linux has become a driver of complex computational medical research, notably within the National Institutes of Health with its Linux-based HPC cluster, critical to handle data-sequencing initiatives. Large-scale genomic sequencing pipelines stemming from the Human Genome Project, and structural neuroimaging suites such as the widely used FMRIB Software Library and FreeSurfer, rely natively on Linux environments to process vast datasets of brain scans and genetic maps.[43,44,45] This high-performance computing requirement, combined with an ecosystem of tools, has made the kernel particularly well suited in the healthcare research context.

Today, the Linux kernel is the central processing engine for modern operational and diagnostic technology that define so much of the healthcare industry. Its impact is felt in high-resolution medical imaging clusters and complex laboratory analyzers, as well as in the small, embedded systems running inside wearable patient monitors and drug delivery pumps. Linux provides the compute layer that keeps healthcare devices connected, functional, and with help from the ELISA community, safety certified.

Kate Stewart and Tim Bird speaking at the Embedded Open Source Summit, Seattle, April 2024.

Industrial manufacturing and civil infrastructure

Safety and stability is also a critical attribute of the kernel when it comes to industrial manufacturing and civil infrastructure. Linux provides the critical processing layer that bridges heavy machinery with enterprise networks, including programmable logic controllers, supervisory control and data acquisition (SCADA) systems, edge computing gateways, and smart sensors on the factory floor. The extreme lifecycle of physical manufacturing infrastructure, where equipment is expected to operate safely and continuously for up to forty years, presents a severe technical mismatch with typical software support lifecycles. The Civil Infrastructure Platform (CIP) project was born specifically to address this industrial need, establishing a trusted, collaborative framework to maintain the Linux kernel over decades without sacrificing system stability or introducing proprietary fragmentation.

To solve the longevity gap, the CIP project selects specific Long-Term Support (LTS) Linux kernels and transforms them into Super Long-Term Support (SLTS) kernels. While standard LTS kernels are maintained for only a few years, CIP commits to securing and supporting its SLTS releases for a remarkable 10 to 20 years. Industry participants like Siemens, Toshiba, Renesas, Texas Instruments, and Cybertrust actively co-govern this framework, embedding SLTS as the core engine of their internal operating systems. This shared baseline allows enterprises to deploy a single, highly stable software foundation across diverse product lines with the benefit of simplifying multi-decade maintainability.

Beyond multi-decade support, these physical environments demand microsecond-level determinism, a need addressed by the Real-Time Linux (RT-Linux) Project. Through the long-standing efforts of the community, RT-Linux's core PREEMPT_RT patch set was fully mainlined in Linux 6.12. As a key member organization, the CIP incorporates RT-Linux capabilities into its SLTS baselines, ensuring that time-critical infrastructure equipment operates with hard real-time predictability alongside enterprise-grade stability.

The reliability and real-time responsiveness demanded by these real-world physical use cases is facilitated by an automated testing pipeline built on the KernelCI framework. Originally started by Linaro, and later joined by BayLibre and Collabora, this framework expanded into a full foundation of its own, focused on kernel testing infrastructure. Because a failure in critical municipal or industrial infrastructure is an untenable physical risk, the CIP Testing Working Group uses KernelCI to continuously validate SLTS kernels against actual industrial reference hardware. This automated testing environment ensures that backported security patches and bug fixes do not introduce regressions or break legacy systems. The process is rigorous, but necessary, and has paved the way for compliance with stringent global security standards like IEC 62443 and the EU Cyber Resilience Act.[46]

Under the CIP and KernelCI communities, the civil infrastructure and industrial manufacturing industry has evolved a sophisticated, technical discipline. Their work feeds directly into a highly efficient, upstream-first virtuous contribution loop. Under this methodology, CIP developers first submit all bug fixes and technical enhancements directly to the primary, mainline Linux kernel codebase. Only after these changes are vetted and merged by the broader open source community are they backported into the SLTS releases. This approach not only prevents proprietary fragmentation and significantly reduces the total cost of ownership for member companies, it also creates a cycle of mutual benefit.

By active engagement in testing, validation, and upstreaming, CIP's specialized working groups improve the stable kernels and Debian baselines, making the broader Linux ecosystem more secure and resilient for the industrial manufacturing and civil infrastructure industry.[47] These communities are among the longest running open source projects under the Linux Foundation umbrella. KernelCI started in 2014, and the CIP project in 2016. Together, they illustrate a pattern: As the Linux kernel proliferated, so too would more specialized forms of industry-specific open source collaboration.

> **As the Linux kernel proliferated, so too would more specialized forms of industry-specific open source collaboration.**

Space, drones, and autonomous systems

In the aerospace and autonomous flight sectors, the Linux kernel has made a unique mark. For decades, aerospace architecture relied almost exclusively on expensive, radiation-hardened microcontrollers running highly constrained real-time operating systems. However, much like in the healthcare applications described previously, the advent of modern space missions demanded computing infrastructure that far exceeded legacy hardware boundaries. Space demands real-time computer vision, autonomous landing calculations, and high-throughput data processing, and the Linux kernel was ideally suited. Embedded Linux offered an open, highly adaptable platform capable of running complex algorithms on multi-core processors. This enabled space agencies and private aerospace pioneers to transition from basic telemetry to fully autonomous operations for inner- and outer-space environments.

A landmark moment in this evolution was NASA's Ingenuity Mars Helicopter, which made history in 2021 by achieving the first powered, controlled flight on another planet.[48] Running embedded Linux on a commercial-off-the-shelf processor as its primary onboard computer, alongside JPL's open source F´ (F Prime) flight software framework, Ingenuity demonstrated that open source software could survive and excel in extreme extraterrestrial environments.[49] Operating with a roughly 15-minute communications delay from Earth, the helicopter relied on Linux to execute its visual-inertial navigation algorithms in the thin Martian atmosphere, and analyze sensor data and camera frames in real time to maintain stability.[50] This success marked a new paradigm, proving that low-cost hardware running Linux could handle mission-critical performance loads in harsh, remote environments.

Beyond planetary exploration, Linux underpins low-Earth orbit constellation networks, cargo transport craft, and satellite systems. On the International Space Station, primary laptops and command systems run Linux distributions to ensure operational reliability and immunity to system crashes during critical experiments.[51] Similarly, commercial spaceflight leaders use Linux across vehicle avionics, booster recovery stages, and satellite internet constellations.[52] Starlink is the largest deployment of Linux in space, with 32,000 highly internetworked satellites.[53] To further accelerate innovation within the space industry, a new incubating project, **Space Grade Linux**, is currently working on a standardized base layer of Linux for use in a variety of space contexts, so that the industry can move in lockstep.[54]

That transition did not begin in orbit. For two decades, the same capabilities have been maturing closer to the ground, where Linux became a standard part of the modern drone stack. A typical autonomous drone pairs a real-time flight controller

Source: *By Andrea Luck - Ingenuity - The First Epic Mars Helicopter - Nasa's Ingenuity - Sol 44, CC BY 2.0,* **https://commons.wikimedia.org/w/index.php?curid=145282930**

with a Linux companion computer that carries the aircraft's higher-level autonomy, from computer vision and obstacle avoidance to connectivity and fleet management, increasingly alongside the open source middleware **Robot Operating System 2**. That architecture is standardized in the open by the **Dronecode Foundation**, the Linux Foundation project that stewards the PX4 autopilot and the Pixhawk open hardware standards. Autonomous air taxis push the same design further, using Linux for sensor fusion across LiDAR, radar, and optical cameras. Virtually the entire drone development lifecycle, from simulation to continuous integration to flight log analysis, runs on Linux.

Because autonomous flight is a safety-critical domain under strict aviation oversight, aerospace engineers collaborate through open source initiatives like the previously referenced ELISA project, which Boeing joined as a Premier member in 2022 to help launch an Aerospace Working Group. Linux already flies in systems certified at Design Assurance Level D, PREEMPT_RT in the mainline kernel supplies the determinism flight applications demand, and the community is methodically building the assurance case for more critical roles. Even the certification tooling is developed in the open: a University of Illinois and Boeing structural coverage tool that measures the kernel against DO-178C requirements earned Best Paper at the 2025 Digital Avionics Systems Conference.

"Linux won in drones quietly, on merit, and by one strong decision at a time. Autonomous systems in space call for the same discipline and confidence. One shared baseline, built in the open, establishes the trust required to create and scale autonomy here on earth and all the way to the stars," said Ramón Roche, Executive Director of Dronecode Foundation.

Linux Kernel Developer Series: Julia Lawall

Julia Lawall's developer background originates in academic research on programming languages and program transformation. After earning her PhD in the early 1990s and spending years in academic programming-language research, she decided to transition into operating systems and kernel research around 2000. In 2004, during the transition between Linux 2.4 and 2.6, Julia and her research team spent over a year manually analyzing hundreds of kernel API changes, finding that when people make changes in an interface, those changes have to be reflected manually across all device drivers, and this refactoring was error-prone. They set out to automate API updates for device drivers across kernel versions.

Julia and her colleagues Gilles Muller, Yoann Padioleau, and René Rydhof Hansen developed Coccinelle, a semantic patching tool designed specifically to allow developers to express structural C-code transformations easily using patch syntax. They started to send out small patches on relatively obscure errors, which led to a big patch of cross-subsystem development with over 100 updates. This patch received pushback from maintainer Andrew Morton, who took issue with lumping the updates together. At the time, the MAINTAINERS file did not have enough information to easily identify which maintainer should receive which update, so he agreed to carry the patch forward. Ultimately Coccinelle became a staple tool within the Linux kernel, and to date, it has been cited in over 10,000 kernel commits. Julia said, "The thing that makes Coccinelle work is that the kernel, for all of its diversity, has a lot of commonality. And we also have this very strong review culture where common criteria are applied... The structure of the code looks the same, the major APIs look the same. And so that's what's perfect for something like Coccinelle."

Today, Julia serves as a senior researcher at Inria in Paris, maintains Coccinelle, and serves on the Linux Foundation Technical Advisory Board. She has mentored for Outreachy and LFX Mentorship programs and focuses her current research on scheduler performance and formal software verification. On why she chooses to stay in the community, Julia said, "this is a really great community... We've got this huge wonderful resource in front of us, this huge codebase, which is very diverse and very interesting and you can get really into any part of it. Each part is a whole new exciting world of things to understand how things fit together."

Listen to Julia's full interview, **My Life as a Linux Kernel Developer and Maintainer with Julia Lawall** [55]

Linux's technological impact

The history of Linux is one of significantly outmatching the quality and utility of existing proprietary technologies, open sourcing it, and thus democratizing access to cutting-edge solutions.

> **From its inception, Linux has expanded and evolved to meet the technological moment, at once influencing and being influenced by new operating structures, softwares, and use cases.**

The following subsections illustrate how Linux has evolved to meet the latest computing needs and how these computing ecosystems have embraced and evolved around the Linux kernel.

Personal and enterprise computing

Enterprises quickly learned that using Linux for data processing and networking would save them money while providing stability, customizability, and scalability. Red Hat founders saw Linux's customizability as a key differentiator for enterprises, and started selling Linux as an alternative to Unix, not for "being better, faster or having more features," but for this singular feature of adaptability.[56] In the late 1990s, developers at IBM started experimenting with Linux and published patches for it to run on its S/390 mainframes.[57] The release of Linux 2.2.13 at the tail end of 1999 made the kernel compatible with mainframes to handle enterprise workloads and large I/O throughput; today, it runs most modern workloads.[58] Beyond mainframes, companies started to recognize that running Linux—alongside Apache, MySQL, and programming languages like Python—on a cheaper PC could give them a world-class server to scale up their internet business. Linux became increasingly popular in server rooms in the era of the consumer internet.[59] Today, Linux powers servers for Google, Amazon, Netflix, Uber, CERN, major stock exchanges, and more. More than 90% of the Fortune 500 use Red Hat solutions,[60] and Linux is the predominant operating system of all websites.[61]

Linux also disrupted the workstation market by providing an environment for high performance on a standard PC. Instead of having to buy an expensive workstation machine, you could buy a PC and install Linux to achieve the same level of performance. Industries with significant computing needs started to take note. The automotive and film industries were trailblazers, with companies such as DreamWorks, Pixar, and Volkswagen implementing Linux workstations in the early 2000s.[62,63,64] Linux is also the favoured system for workstations in highly regulated industries, such as defense, government, and financial services, thanks to the security, stability, and long-term support guarantees in Red Hat's workstation license.[65]

As desktops became mobile with the increased use of laptops and cell phones, Linux had to learn how to save power. Originally designed to operate with a data center's supply of energy, the Linux kernel had a hardware timer that forced the CPU to wake up and check for tasks to run hundreds or

thousands of times per second. In 2007, as demand grew for better battery power of mobile devices, developers started to design for more efficient power management and the Tickless Kernel was developed, where the kernel checks for events and the timer is programmed to fire when the next event comes due.[66]

Mobile computing has greatly increased the global OS usage of Linux. At first blush, it appears to have only minimal use, showing around 3% of desktop downloads for computers and laptops.[67] However, the high-selling Chromebook has in fact made Linux much more prevalent: the browser-based ChromeOS is fundamentally a Linux distribution, meaning those using Chromebooks are using Linux.[68] Beyond laptops, the use of smartphones also contribute to Linux's market share in personal and work computing, as Android makes up 72% of the mobile OS market share globally, with upwards of 4 billion Android devices in use.[69]

Linux's influence on computing comes from more than its cost-effective and modular architecture. The open source community around the kernel became a strategic business decision for mainframe computing that continues to resonate today. As Irving Wladawsky-Berger, a former IBM executive who led the Linux initiative, has explained, "IBM did not embrace Linux [in 2000] because of the Linux kernel, which at the time we knew needed a lot of work. But we looked at the Linux community, really liked what we saw, and that gave us confidence that Linux would get much better over time, especially if companies like IBM contributed to its development, which we did."[70]

As Sasha Levin, kernel maintainer, explained in an interview, more and more companies started to align with the upstream community as Linux became the defacto standard for operating

Jim Zemlin speaking at Open Source Summit North America, Minneapolis, May 2026.

systems in the server and data center world. "There is no choice but to be a good citizen of the community. It was painful to have those out-of-tree drivers... to chase the progress that the kernel is making. Painful not only as far as technology goes, but painful also as far as the economics go—you have customers that use Linux and they expect to be able to use [your] products on their kernel."

Linux Kernel Developer Series: Shuah Khan

Shuah was a long-time Linux user before making her first upstream open source contributions as an independent contributor. Her entry point into kernel development came during the Android mainlining effort, when code from Android git repositories was brought into the mainline kernel. Her very first contribution involved removing an Android staging driver, a patch which was accepted by Greg Kroah-Hartman. She continued with Android features, quickly learning the LED driver framework, how different triggers work, and how to test driver behavior, using her personal laptop. "You have to have a different mindset to work in open source," she explained. For new contributors, she explained, "You have to... have the courage to send your code out and get it reviewed by a lot of people... [you are] seeking input from others that know the kernel better than [you] do... don't get so attached to the first idea you implement."

Shuah's open source experience expanded through various corporate and open source roles. She worked on ProLiant workstations at Hewlett-Packard, where she supported Linux distributions Red Hat and SUSE by debugging complex interactions between firmware and the kernel and identifying backports to support 2TB memory configurations and IOMMU, among other things. At Samsung, she was one of the first engineers hired into the open source group, where she served in a role that bridged product team needs with upstream kernel features, maintaining development boards across kernel releases, and leading internal mentoring classes. Shuah started supporting stable releases and took over maintainership of Linux Kernel Selftests, transforming a scattered handful of tests into a unified, framework-driven testing suite connected to the main kernel Makefile. She shifted the community mindset, organically making test creation a default part of feature development. Drawing on her background as a mentor and mentee, she built and grew the Linux Foundation Kernel Mentorship Program from a pilot of 2 mentees into a global initiative training around 100+ developers each year using real-world tasks that they choose to focus on. When assessing the success of the mentorship program, she said, "I don't look at outcomes... Instead, I am empowering them to make their dreams come true... It is a vehicle to empower [mentees] to find their path, whatever that might be."

Listen to Shuah's full interview, **My Life as a Linux Kernel Developer and Maintainer with Shuah Khan** [71]

Cloud computing and containers

The evolution from physical to virtual servers took an increasingly interrelated path with the Linux kernel. Resource inefficiencies, scalability issues, and basic space needs from running applications on physical servers led to the more mainstream development and adoption of virtualization.[72] At first, developers started running virtual machines that emulated the underlying hardware; containers were then developed to virtualize the operating system instead of the hardware, packaging application code with all of its libraries and dependencies.[72]

Linux evolved over time in order to effectively deploy virtualization and containers. As of late 2014, the Linux kernel includes three important features to support modern containers: Namespaces, which restrict what each of the applications running in the container can see, providing an isolated network stack, IP address, and file tree; control groups, which monitor and manage how much an application can consume of the CPU, memory, disk I/O, and processes; and OverlayFS, a file system that appears unified to the application but maintains the underlying container image while only modifying a temporary top layer of the system.[73] Namespaces were introduced to the kernel over a period of time (2002-2013), while control groups were merged into the main kernel in 2008 and OverlayFS at the end of 2014.[74,75,76,77] This last contribution, released under version 3.18, meant that the kernel became a self-contained native container platform. Before that release, building an enterprise platform or running Docker meant manually patching your kernel or using slow storage drivers. Today, approximately 90% of public cloud infrastructure is run on Linux, and it powers 49.2% of global cloud workloads.[78,79]

Linux's dominance in the virtualization market has meant that it has, in turn, caused the cloud community to evolve around its significance. Multi-billion dollar companies and solutions have cropped up to support Linux-based containers: for example, the Docker platform was created to simplify the complexity of creating a container by translating human-readable commands into low-level requests for the Linux kernel to execute. Projects as well have sprung up to support various aspects of cloud computing, such as eBPF, which allows for observability in packet filtering at the kernel level, reducing complexity and increasing security.[80] Because of the ubiquity of Linux at both the cloud and mainframe level, the Open Mainframe project enables a hybrid cloud approach, aligning both computing environments around the Linux OS, standardized containerization, and common DevOps pipelines.[81]

"The cloud native movement owes its existence to core primitives engineered directly into the Linux kernel, most notably cgroups, namespaces, and eBPF. What the industry calls 'the container revolution' was ultimately the creation of orchestration frameworks like Kubernetes and Docker built on top of Linux's native kernel isolation and scheduling capabilities. At 35, Linux remains the indispensable runtime of modern software workloads," said Chris Aniszczyk, CTO, Cloud Native Computing Foundation (CNCF).

Twenty years to “Real Time” aka PREEMPT_RT

In September 2024, at a conference in Vienna, Thomas Gleixner handed Linus Torvalds a printed pull request.82 Linus merged it, and Linux 6.12 became the first mainline kernel with built-in real-time support. The moment closed a twenty-year campaign to make Linux a true real-time operating system.[83,84]

The PREEMPT_RT effort began in 2004 with Gleixner, Ingo Molnar, Steven Rostedt, and the late Doug Niehaus, who set out to give Linux bounded, predictable response times. The late Daniel Bristot de Oliviera also contributed significantly, particularly in the deadline scheduling code. In the final stretch, Linutronix engineers Sebastian Andrzej Siewior and John Ogness carried much of the load. Ogness resolved the last major blocker by reworking the kernel’s printk logging so console output could no longer stall real-time work.[85]

Source: *“The realtime preemption pull request.” LWN.net. Available from:* **https://lwn.net/Articles/990985/**

Code was never the hard part. Funding two decades of deep infrastructure work was. Germany’s OSADL organized early industry support. In 2015, the Linux Foundation launched the Real-Time Linux collaborative project,[86] named Gleixner a Linux Foundation Fellow, and pooled member funding to finish the upstreaming. Members over the life of the project included Altera, ARM, BMW, CIP, ELISA, Ericsson, Intel, IBM, Google, National Instruments, OSADL, Red Hat, Siemens, and Texas Instruments.

The effort resulted in one Linux codebase that runs from cloud servers to microsecond-critical devices. Commercial offerings such as Red Hat Enterprise Linux for Real Time, SUSE Linux Real Time, and Ubuntu’s real-time kernel now serve industrial robotics, software-defined vehicles, 5G radio networks, medical devices, aerospace and defense systems, and professional audio. With the ELISA project advancing safety certification, real-time Linux is positioned for the next wave of robotics and physical AI at the edge.

Real-time Linux is also a mentorship story. Anna-Maria Behnsen joined the Linutronix real-time team early in her career, learning the kernel with her father, Thomas Gleixner, as a mentor. She grew into one of the kernel’s timing experts, rearchitected how the timer subsystem handles expiry across CPUs, and in 2024 was named co-maintainer of the kernel’s high-resolution timers and timer wheel.[87] The code that keeps Linux on time is now a second-generation responsibility.

Edge

In an edge computing system, data is collected from diverse physical systems and then is processed locally to avoid sending everything to a cloud. This hardware diversity and localized, real-time processing requires the modularity, determinism, and containerization of the Linux operating system.[88] The Linux kernel offers hardware portability, microsecond-perfect execution times, cloud native on constrained hardware, and scaling through open licensing.

> **Today, more than 75% of edge devices around the world run on the Linux kernel.[89]**

Linux has played an important role in advancing edge computing. Before "edge" existed, embedded systems developers turned to Linux to avoid royalties on proprietary, specialized real-time operating systems (RTOS). Using Linux, they could strip away unnecessary drivers in order to build lightweight, customized kernels.[90] The popularity of Linux for embedded systems yielded a wider ecosystem of open source projects, such as the **Yocto Project** and **LF Edge**. As the kernel evolved for containerization, this gave developers the tools to run application software identically no matter whether the hardware was on a laptop, a cloud server, or an industrial gateway.

As the edge computing ecosystem evolved around Linux, the kernel equally evolved to meet the needs of edge computing. For example, the upstream merge of PREEMPT_RT enabled industries to rely on the quality processes associated with the Linux kernel and simplify the analysis when used in critical applications that required determinism. This has increased Linux adoption in latency-sensitive edge environments such as medical devices, and it became the fastest-growing revenue share for edge in healthcare in 2025.[89] Operating systems for the edge have also emerged, such as Ubuntu Core and Alpine Linux that are optimized to run on minimal, low-resource systems.[91] For larger edge infrastructure, such as cell towers and regional data centers, tools like eBPF in Linux allow local edge nodes to inspect and route high-throughput traffic at speed and bypass the typical processing overhead.[92]

Edge compute has evolved across IoT, enterprise, telecom and cloud edge. Rather than replacing the Linux kernel, projects (EdgeX, EVE, Open Horizon, Akraino) extend its capabilities, creating an open, vendor-neutral ecosystem that accelerates innovation across industrial IoT, retail, manufacturing, smart cities, telecommunications, and AI. As edge infrastructure increasingly supports AI inference and real-time analytics, the Linux kernel continues to serve as the trusted runtime that connects heterogeneous hardware with cloud native software, making it the cornerstone of the open edge computing ecosystem.

Linux Kernel Developer Series: Mark Brown

Mark Brown's entry into Linux began 30 years ago as a Debian developer. His earliest kernel contributions were in the networking stack, where he upstreamed performance fixes and developed a workaround for a severe hardware bug on a network controller. He started on part-time upstream kernel work as part of his company's embedded products before he moved to full-time kernel development at Wolfson Microelectronics, a chip manufacturer, where he wrote and maintained driver code for Wolfson devices. He later moved to Linaro and then Arm, always maintaining an upstream-focused role.

Mark is a maintainer across several subsystems, and contributes to ALSA, testing, and regmap. Mark also manages Linux Next, the main integration tree for testing patches prior to mainline merge, following Stephen Rothwell's retirement. In this role, he is witnessing the growth of LLM-assisted patches and how these are greatly increasing maintainer workloads. He describes how trust in these tools is under question, despite the community's familiarity with AI-assisted maintenance: "things like Coccinelle make it really easy to do widespread code transformations... but those are all deterministic and regular... With LLMs, they're a lot less predictable... you can't have quite the same sense that 'yes, I trust this tool, this tool will probably do the right thing.'" Beyond code contributions, he serves as the organizing committee chair for the Linux Plumbers Conference.

As to why he chooses to stay involved, he said, "it's a combination of the technical challenges and trying to make something that does the best possible job for people. The people we're working with is a key part of it."

Listen to Mark's full interview, **My Life as a Linux Kernel Developer and Maintainer with Mark Brown** [93]

Artificial intelligence

Modern artificial intelligence was born on the Linux kernel. Linux provides the essential high-throughput, low-latency foundation required to power new AI workloads. AI tools and automated agents are simultaneously transforming the developer practices, security maintenance, and code review workflows of the kernel itself.

As the foundation for global workload infrastructure, the kernel is experiencing a rapid telemetry-driven modernization to support the highly volatile and parallelized nature of machine learning workloads. For example, the sched-ext framework introduced in 6.12 allows for hyper-specialized, programmable schedulers for many types of workloads, including AI. This framework allows workloads to bypass the rigidity of Linux's one scheduler and instead run a customized policy as a safe BPF program that allows for rapid experimentation and reduces bottlenecks at the CPU level.[94,95] AI agents are also forcing more flexibility into the kernel to manage their erratic resource usage throughout a workflow, using minimal memory to read code files but massive usage when it executes a call.[96] New frameworks, such as AgentCgroup, allow for flexible memory allocation in the kernel, with temporary and granular resource budgets.[96,97] These new frameworks help the kernel to stretch dynamically based on real-time execution telemetry.

The computing needs of AI workloads has also forced the semiconductor industry to pivot from isolated hardware manufacturing to architectural co-design with the Linux kernel and the open source ecosystem. The fact that AI agents have to call tools, retain information in memory, create sub-agents, and make decisions means that no single processor can handle their workflows. As a result, they require specialized hardware orchestrated through codesign, where manufacturers

optimize their chip architectures simultaneously with the operating system kernel.[98,99] Relatedly, as the requirements of AI workloads diverge from general-purpose processor design, chip manufacturers are incorporating open-standard Instruction Set Architectures such as RISC-V that enable the development of processors tuned to specific, customized workloads at a lower cost.[100]

> **The chip ecosystem is synchronizing to meet the compute demands of generative AI, with manufacturers shifting from proprietary hardware development to open source Linux kernel integrations and open-standard architectures to ensure their silicon can scale across global cloud networks.**

Looking beyond the technical requirements of AI, the kernel has also had to evolve to reckon with the impacts of AI tooling. Along with every other corner of software engineering, the Linux kernel has received more and more AI-assisted code patches, and the community released a policy in April 2026 on the use of AI tooling, encoding its use as an assistive tool that requires attribution and a human sign-off.[101] The policy was created for better visibility into AI-assisted patches, but managing the flood of patches is an ongoing concern, as they clog the development pipeline with often duplicate and redundant reports.[102] On the maintainer side, AI tooling for code reviews has gained traction in the kernel community. One example is the Sashiko project, an agentic tool that ingests patches and evaluates them based on a review protocol that mimics a team of reviewers.[103] In July 2026, Torvalds confirmed the kernel's use of AI tools, in reference to Sashiko, expressed in a mailing list: "Linux is not one of those anti-AI projects… AI is a tool, just like other tools we use. And it's clearly a useful one."[104] As AI tooling evolves, the kernel's response to it will undoubtedly continue to influence other project policies and practices, creating a ripple effect in the adoption of AI for programming and bug fixes.

As with all open source projects, AI's expansion of the threat landscape has called for a reevaluation of the kernel's security. Malicious actors use AI tools to crawl public code bases and identify bugs that can be chained together to create complex exploits.[105] In response to this, security maintainers are using the same tools defensively, finding where a pattern of bugs have been left unaddressed and generating patches and bug fixes. Enterprise stakeholders are banding together to defend against this new threat landscape, such as in IBM and Red Hat's Project Lightwell and the Linux Foundation's Akrities.[106,107]

Linux Kernel Developer Series: Sasha Levin

Sasha Levin describes his path into open source software development as beginning when he was tasked with writing a networking driver that operated on both Windows and Linux. While the work with Windows required sifting through documentation and searching for workarounds, the Linux side allowed him to look directly at the source code, identify problems, and send fixes back. This experience led him to transition into full-time Linux kernel development, and he started building the KVM tool (a lightweight hypervisor) roughly 15 years ago. Over time, he gravitated toward core kernel work, live patching, and maintaining the Linux stable releases. Additionally, Sasha has been applying automated tools to the kernel long before generative AI became popular, using these tools to identify and mainline fixes into stable releases. As he said, "I think AI can be a great tool to understand code, and understand why things are done in a certain way."

In his current role at NVIDIA, he leads safety certification efforts for the Linux kernel and pushes those features back upstream. For Sasha, finding the right relationship and role within a company has been critical for managing his maintainer workload, as he explained: "I really wanted that my work on upstream would not be seen as a donation or a charity that I'm doing to open source. I wanted it to be beneficial to both the company, to myself, and to the kernel world... It's a matter of structuring your career around what you love doing."

Listen to Sasha's full interview, **My Life as a Linux Kernel Developer and Maintainer with Sasha Levin** [108]

Sustainability and the future of the kernel

With this incredible impact, an obvious question persists in the Linux community: How do we ensure the kernel's sustainability and future? Now 35 years on, there are signs of a very healthy open source project: there are hundreds of new contributors every release cycle and over half of developers continue to contribute quarter over quarter (see **FIGURES 1 AND 3**); the code base is also growing and shrinking, indicating that features are being removed as much as added. However, new threat landscapes imposed by AI and the ever-important maintainer pipeline mean the community has to continue to put sustainability at the forefront of its strategic planning.

Security

The ubiquitousness of the kernel means that its security has very significant implications across the global tech stack. In 2015, the Linux Kernel Self-Protection Project (KSPP) was launched to proactively integrate structural defenses into the Linux kernel. Its goal is to engineer the Linux kernel to fail safely when unknown vulnerabilities are triggered. For example, KSPP spearheaded the systemic eradication of the notoriously unsafe strcpy function (which blindly copies data until it hits a marker, risking catastrophic memory corruption), replacing with bounded alternatives like strscpy that strictly limit the data transferred. Similarly, the project modified the build system to reject accidental "drop-throughs" in logical case statements. In the past, a developer could forget a simple break instruction and cause the processor to accidentally execute the wrong block of code. Today, developers must explicitly mark any intentional drop-through, preventing a dangerous accidental omission. Ultimately, these self-protection mechanisms ensure that whole error categories are eliminated before the software ever runs.

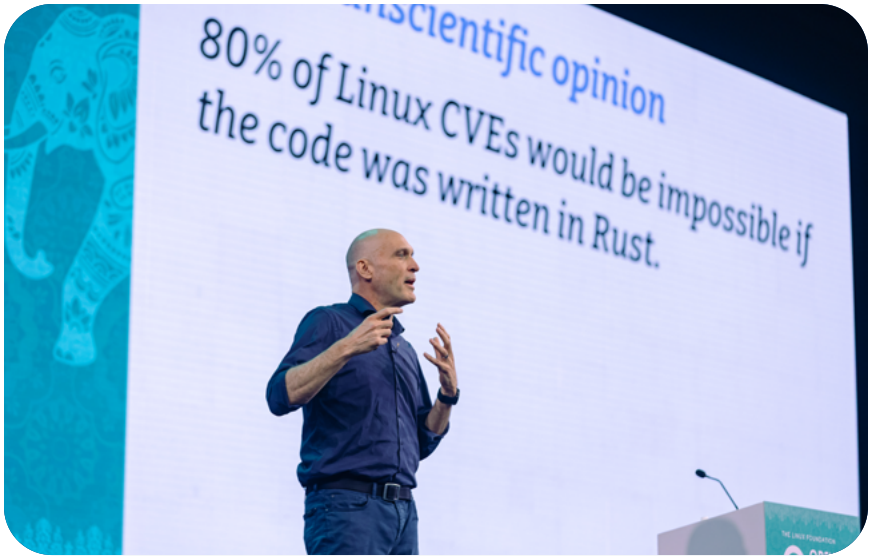
The memory safety crisis of modern computing (the realization that most vulnerabilities in many systems are memory safety errors) urged security-minded community members to take on the significant task of writing parts of the kernel in Rust. Rust was chosen because it guarantees memory safety at compile time through strict type and borrow checking, avoiding the performance penalty of a runtime garbage collector. This use of Rust maintains the performance of C while removing the possibility of entire classes of vulnerabilities being introduced into those portions of the kernel.[109] In 2020, the Rust for Linux project started to test whether the language could coexist with existing C code. The initial Rust infrastructure was merged in 2022 but was considered experimental, considering it had been many decades since anyone had attempted to build a significant part of a production kernel in any high-level language other than C or C++.[110] From there, the first production Rust drivers were merged, and adoption grew, despite some friction in the maintainer community.[111,112] In late 2025, the community announced that Rust was no longer experimental.[113] As Greg Kroah-Hartman has said, the choice to use Rust was driven by the need to diminish the reviewer workload, not necessarily to improve the developer experience.[109]

The Linux kernel has earned the OpenSSF Best Practices Gold Badge, the highest tier of security self-certification for open source software, by meeting rigorous criteria.[114]

Despite meeting this milestone, the kernel must continue to evolve to defend itself against new threats and exploitations. The security of the open source community itself went under scrutiny with the XZ Utils exploit, where a likely-state-sponsored actor targeted the burned-out solo maintainer of XZ Utils, gained their trust, and injected a backdoor that was accidentally caught by an engineer.[115] This emphasized the need to prevent social engineering attacks, and encouraged a shift in governance for solo maintainer projects and stricter access controls.[116] The code review process is also under new pressure as AI-generated patches flood open source software maintainers, including the Linux kernel mailing lists as discussed in the previous section.

This is particularly troubling for the Linux kernel's private security list, which is meant for private reporting to avoid exploitation. This influx of reports is making the list almost unmanageable with numerous redundant or irrelevant security patches. To address this flood, Linus announced that anything that an AI bot is reporting should not be reported in the private security list, since it is probably already public.[117] In the spring of 2026, the community made some updates to the kernel documentation to address these security reports, clarifying what qualifies as a security bug, what a vulnerability is and is not, what should be handled in the regular development process, and how to use AI tooling properly in the private security list.[117]

The Linux kernel developers have learned and taken advantage of the experiences and tooling of others, and in return, have demonstrated how to find and counter vulnerabilities at scale for others. As David A. Wheeler, director of Open Source Supply Chain Security at the Linux Foundation, described, "Thirty-five years of Linux kernel development have proven that security at global scale requires open collaboration and



Greg Kroah-Hartman speaking at Open Source Summit India, June 2026.

uncompromising honesty. Linux maintainers do not hide behind narrow vulnerability definitions or sweep edge cases under the rug. The Linux kernel community carefully examines and re-examines code, conservatively treating many software defects as potential security risks, to help all its users stay secure."

Mentorship for the next generation

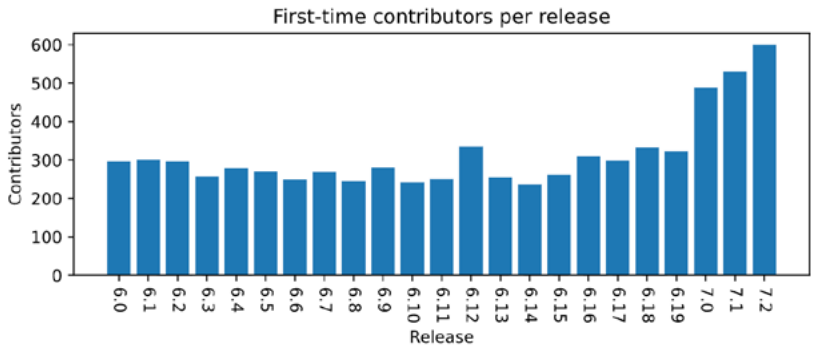
The Linux kernel community functions through a model of voluntary, self-directed participation and interest. Contributor progression follows what Shuah Khan, kernel maintainer and Linux Fellow, describes as a "road," where developer involvement depends on personal agency, interest, and available time. Khan said, "In open source, it's not like you recruit new developers, and those recruits go on to become maintainers. It is a community of people that voluntarily come

and stay, choose to stay or leave. It's not somebody pushing you as a developer through a pipeline. You are in charge of traveling that road."[118] A developer may become a frequent and long-standing contributor, or they may be more short-term and transient, submitting a small number of patches to solve specific problems before moving on. Although continued engagement is ideal to build the community, any form of engagement helps disperse open source methodologies, technical standards, and collaborative practices into the wider software industry.

Building future generations of kernel developers is critical to ensure the longevity of the project. Contributions to the kernel grew dramatically from 1991 and started to stabilize in 2014 between 12-15,000 commits per release cycle contributed by an average of 2,000 contributors. More recently, there have been encouraging signs of a healthy developer community: 2,362 developers contributed changes during the kernel's 7.0 merge, with 489 of those being first-time contributors.[119] This increased number of newcomers continued in the 7.1 and 7.2 merge, and is most likely driven by the use of AI tooling (see Figure 3).[3,120] Today, contributions are heavily skewed toward the United States (25%), with Germany and China coming second and third.[20] Leadership succession is a key aspect of the kernel's continuity, and in January 2026, the community drafted a process for selecting Linus's successor.[121]

The LF Mentorship program is a critical aspect of the kernel community's sustainability and diversity. Truly diverse participation requires addressing systemic barriers to entry, as participation in open source development is unequally distributed across geographic, linguistic, and financial lines. This was a main reason the Linux Foundation established the LF Mentorship Program in 2019, to increase diversity of the open source developer community.[122] The program includes

FIGURE 3: First-time contributors per release, from 6.0-7.2



Source: *"Development statistics for the 7.2 kernel." LWN.net. Available from:* **https://lwn.net/Articles/1088776/** [3]

formal mentorship programs as well as more flexible learning opportunities and stipends to support focused learning.[122,123] The **LF Live Mentorship and Maintainer series** interviews key members of the community on their career paths and tackles specific technical development areas, providing an avenue for knowledge translation without taking up significant hours of experts' or learners' time.[124,125] Beyond the hard skills, the webinars and mentorships teach the social aspect of coding, including how to build credibility, navigate social norms, and establish community trust. From January 2019 to mid-2026, the program has had over 6,447 applicants, from over 111 countries (the significant majority from India), with 73% identifying as male, 14% as female, and 0.8% as nonbinary.[126]

The integration of generative AI represents the latest shift in kernel development, with indications that these tools are a boon to the initial phase of development by generating

boilerplate code, drafting technical documentation, and helping developers navigate complex subsystems. However, relying on automated tools for code creation risks bypassing fundamental software engineering practices and deep semantic comprehension that are essential for long-term success and mastery. The use of AI tools also reduces the human element of mentorship, as Khan explained: "LLMs can teach you the technology, but open source is so much more than technology. 'People' aspects are the difficult ones to navigate and master. Those are the things that you would learn by having that interaction with a mentor."[118] The future strength of the community relies on preserving the channels that teach developers how to navigate open source's human side.

> **Relying on automated tools for code creation risks bypassing fundamental software engineering practices and deep semantic comprehension that are essential for long-term success and mastery.**

This human side is a key element of what keeps Khan engaged in the project: "There's no other project that gives you this kind of fulfillment in terms of being able to work with people from across the world. If you're like me, and you want to do that, then this is the best project."[71]

Linux Kernel Developer Series: Greg Kroah-Hartman

Greg Kroah-Hartman began his own journey as a Linux maintainer in the embedded world while attempting to fix a USB specification issue in the kernel. He started working on the project in his spare time, recalling his initial contribution: "my first time I posted in the mailing list was, 'How do I write a patch?' I didn't know how to write a patch because there wasn't really any good documentation." He explained how working with experienced community members and absorbing their direct feedback made him a better engineer. Having served as a USB maintainer for many years, Greg described kernel maintainorship as distinct from traditional jobs, since maintainers carry their knowledge and subsystem responsibilities with them across employers and roles. He summarized this uniqueness by stating, "we can't just hire maintainers. They have to grow into maintainers".

To manage his diverse workload across various subsystems, Greg discussed the importance of prioritizing his roles: reviewing stable releases early in the week, queuing patches by midweek to send to Linus, advancing work on Rust, and managing USB contributions—and in all that, learning how to step away and delegate. The knowledge built over decades of maintaining the kernel provides the domain experience necessary to recognize critical technical patterns, which is particularly relevant for the community's AI pattern matching work. Reflecting on why he continues to do this work and the satisfaction he gets from it, Greg noted, "it's really easy to make a kernel once. It's really hard to do something over time, constantly, repeatedly, that's stable and works for everybody. That's an amazing engineering feat."

Listen to Greg's full interview, **My Life as a Linux Kernel Developer and Maintainer with Greg KH** [127]

The kernel's human impact

> **It's really easy to make a kernel once. It's really hard to do something over time, constantly, repeatedly, that's stable and works for everybody. That's an amazing engineering feat.**
>
> **– Greg Kroah-Hartman, Linux Fellow**

By providing a highly performative, modular, and free foundation for global computing, the Linux kernel has reshaped economic equality, global collaboration, software governance, and digital sovereignty. Beyond the industry and technical impact of Linux, the kernel and the open source ecosystem has profoundly shaped and enabled everyday human experiences around the world. High-transaction financial systems built on and enabled by open source—such as India's UPI mobile payment infrastructure and PayPal—allow small business owners, independent vendors, and rural communities to execute digital transactions with nothing but a QR code and app on their phone. This powers a significant chunk of India's economy: in June 2025, the UPI processed over 18 billion transactions making up ₹2.4 trillion (about 25 billion USD) in payments.[128] In agriculture, open source controllers allow small farmers to monitor soil moisture and predict local weather patterns. In energy grids, small Linux-driven controllers operate green energy infrastructure, like wind turbines, to manage regional power generation. Open source powers the data centers and servers behind global communication platforms, allowing long-distance families and friends to stay connected and enabling worldwide scientific collaboration during global crises.

In conversation with Shuah Khan, she put it this way: "I think about what the world would have looked like without open source... And what would we be doing without open source?" The distributed and free nature of development has meant new perspectives and ideas, and a form and direction of progress that would not be possible in a closed source environment. She continues, "Linux runs the world, Linux and open source. That's not a statement that we make lightly. And that is the beauty of it: people don't even know Linux exists... but we have an outsized influence on the world." Despite the significant momentum of the ecosystem, there was not an inevitability to an open source operating system, or any other open source projects—they took, and will continue to take, the right people at the right time. This is no less relevant today, as the current tension around open source AI models mirrors the enterprise operating system wars of the early 2000s. This comparison offers a clear historical lesson: an open source option always provides a secure, customizable, and cost-effective pre-competitive baseline that a global ecosystem can build upon.

Conclusion

Thirty-five years after its modest origins as a hobby project, the Linux kernel has irrefutably become the foundational engine of modern global infrastructure. Linux's modular architecture and distributed development model revolutionized software engineering. Its industry impact spans virtually every domain, enabling and powering software-defined vehicles, Hollywood rendering pipelines, low-latency financial exchanges, safety-critical healthcare devices, and mission-critical operations in space. Technologically, Linux evolved from enterprise mainframes and servers to become the indispensable runtime for cloud native containers, high-throughput networking, and edge computing nodes.

Looking forward, Linux is uniquely positioned to influence and accelerate emerging technology such as modern AI. As AI demands unprecedented computational scale, the kernel continues to modernize its core fabric to orchestrate complex machine learning workloads and specialized silicon architectures. Simultaneously, AI tooling and review platforms like Sashiko are reshaping kernel maintenance and bug remediation. Backed by proactive security initiatives like the Rust integration and strict community governance policies, the Linux community ensures that rapid innovation never compromises systemic stability.

When the Linux kernel turned twenty, IBM Fellow Irving Wladawsky-Berger reflected on this milestone, calling on enthusiasts not to lose sight of what makes Linux so significant. He said, "What is so precious about Linux is the community. That's the one thing we can absolutely never lose. Keeping our open ecosystem going is very important, so that we can continue to play a role in addressing some of the most important issues facing our society."[70] His words still ring true today.

Attendees of the Linux Storage Filesystem MM & BPF Summit, May 2026.

Disclaimer

LF Research uses generative AI tooling in varying degrees for research-related tasks including aggregating secondary sources, synthesizing data, supporting report drafting, copyediting, and translation. All AI-generated content is human reviewed and revised as needed.

Acknowledgments

The authors would like to thank the numerous contributors to this report, who provided data, peer reviews, and helpful guidance throughout the report's development. We especially thank the following individuals for their support:

- Chris Aniszczyk
- Mirko Boehm
- Mark Brown
- Gabriele Columbro
- Jonathan Corbet
- Mike Dolan
- Tosha Ellison
- Dirk Hohndel
- Arpit Joshipura
- Shuah Khan
- Greg Kroah-Hartman
- Julia Lawall
- Sasha Levin
- Paul E. McKenney
- Walt Miner
- Ramon Roche
- Steven Rostedt
- Konstantin Ryabitsev
- Kate Stewart
- Linus Torvalds
- David A. Wheeler
- Irving Wladawsky-Berger

About the authors

ANNA HERMANSEN is a Senior Researcher & Ecosystem Manager for the Linux Foundation, where she leads research projects and supports end-to-end management of the Linux Foundation's research. She has conducted qualitative and systematic review research in health data infrastructure and open source AI, and has presented on this research work at conferences and working groups. Her interests lie at the intersection of AI, precision medicine, and health data sharing. She is a generalist with experience in client services, program delivery, project management, and writing for academic, corporate, and web user audiences. Prior to the Linux Foundation, she worked for two different research programs, the Blockchain Research Institute and BC Cancer's Research Institute. She received her Master of Science in Public Health and a Bachelor of Arts in International Relations, both from the University of British Columbia.

HILARY CARTER joined the Linux Foundation in 2021 to launch and lead LF Research, established to deliver empirical insights into open source trends, opportunities, and challenges. Prior to joining the Linux Foundation, Hilary was the managing director of the Toronto-based Blockchain Research Institute, a global, syndicated think tank focused on blockchain technology. She has contributed to nearly 200 research projects focused on open source innovation and its adoption across industries. Hilary holds a Master of Science in Management from the London School of Economics, and is a dual Canadian and Irish citizen.

DANIEL M. GERMAN is a professor of Computer Science at the University of Victoria. His research focuses on software engineering, particularly software evolution, open-source software, the impact of intellectual property on software engineering, and the social and technical processes that shape how open-source software is developed, maintained, and governed. More recently, he is examining the impact of generative AI and AI coding agents on open-source software development, with particular attention to how they change the economics of contribution, code review, and maintainership, as well as questions of accountability, provenance, and governance in open-source projects. He is also an active open-source contributor and develops and maintains software tools, with a particular interest in programmable development environments and tools that augment how developers interact with software systems.

References

1. Sandberg KD. The Value of Open Data on Global Entities [Internet]. The Linux Foundation; 2026 [cited 2026 Aug 27]. Available from: https://www.linuxfoundation.org/research/open-data-on-global-entities
2. Digital Applied [Internet]. 2026 [cited 2026 Aug 27]. Mobile OS Market Share 2026: iOS vs Android Statistics. Available from: https://www.digitalapplied.com/blog/mobile-os-market-share-2026-ios-vs-android
3. Corbet J. Development statistics for the 7.2 kernel. LWN.net [Internet]. 2026 Aug 17 [cited 2026 Aug 27]. Available from: https://lwn.net/Articles/1088776/
4. [MAINTAINERS SUMMIT] Coping with the new-developer flood [Internet]. [cited 2026 Aug 27]. Available from: https://lore.kernel.org/ksummit/CANiq72m8se6AYJu_qDc1T=nCEank25nAu2ooZSox_1VQg7SOtQ@mail.gmail.com/T/#m05b386da142a5ce647555b847ae6e7e9a3d0ba75
5. CNCF [Cloud Native Computing Foundation]. Keynote: There's Nothing to Fear About the EU's New Cybersecurity Law - Greg Kroah-Hartman (ASL) [Internet]. 2025 [cited 2026 Aug 27]. Available from: https://www.youtube.com/watch?v=LOT3BccrhpA
6. Martin B. opensource.com [Internet]. 2016 [cited 2026 Aug 27]. What the history of open source teaches us about strategic advantage | Opensource.com. Available from: https://opensource.com/open-organization/16/10/history-open-source-teaches-strategic-advantage
7. TED. The Mind Behind Linux | Linus Torvalds | TED [Internet]. 2016 [cited 2026 Aug 27]. Available from: https://www.youtube.com/watch?v=o8NPllzkFhE
8. Wirzenius L. The early days of Linux. LWN.net [Internet]. 2023 Apr 12 [cited 2026 Aug 27]. Available from: https://lwn.net/Articles/928581/
9. Welcome to LWN.net [LWN.net] [Internet]. [cited 2026 Aug 27]. Available from: https://lwn.net/
10. Çintay E. The Evolution of Linux Kernel Development: A Journey Through Time. Medium [Internet]. 2024 Feb 22 [cited 2026 Aug 27]. Available from: https://medium.com/@Chinacolt/the-evolution-of-linux-kernel-development-a-journey-through-time-01dd2846ada6
11. Linux Kernel Newbies [Internet]. 2024 [cited 2026 Aug 27]. Linux_6.12 - Linux Kernel Newbies. Available from: https://kernelnewbies.org/Linux_6.12
12. Naresh IT [Internet]. [cited 2026 Aug 27]. Linux Architecture Explained Simply Core Guide 2026. Available from: https://nareshit.com/blogs/linux-architecture-explained-simply-core-guide

13. History of Portability in Linux [Internet]. [cited 2026 Aug 27]. Available from: https://litux.nl/mirror/kerneldevelopment/0672327201/ch19lev1sec1.html

14. The Linux Foundation [Internet]. 2026 [cited 2026 Aug 27]. My Life as a Linux Kernel Developer and Maintainer with Jonathan Corbet. Available from: https://zoom.platform.linuxfoundation.org/webinar/register?project=tlf&id=oXytj4iRU1Zir0hQxx0BRBbjPFpAlAGbuuczJizxsoa0x%2B3FIblVzHiL1Ec8NN2xpBjrTne5L3NZbgO3nuE%3D&logo=thelinuxfoundation-color.svg&zoomID=97950815459

15. cregit. cregit/cregit [C] [Internet]. 2026 [cited 2026 Aug 27]. Available from: https://github.com/cregit/cregit

16. LWN.net [Internet]. 2002 [cited 2026 Aug 27]. linux-2.5.4-pre1 - bitkeeper testing. Available from: https://lwn.net/2002/0207/a/2.5.4-pre1.php3

17. Developer Certificate of Origin [Internet]. [cited 2026 Aug 27]. Available from: https://developercertificate.org/

18. git [Internet]. [cited 2026 Aug 27]. Git - A Short History of Git. Available from: https://git-scm.com/book/ms/v2/Getting-Started-A-Short-History-of-Git

19. Blau T. Git turns 20: A Q&A with Linus Torvalds. The GitHub Blog [Internet]. 2025 Apr 7 [cited 2026 Aug 28]. Available from: https://github.blog/open-source/git/git-turns-20-a-qa-with-linus-torvalds/

20. LFX [Internet]. [cited 2026 Aug 28]. The Linux Kernel Organization Insights. Available from: https://insights.linuxfoundation.org/project/korg?timeRange=past365days&start=2025-08-28&end=2026-08-28

21. Hoffmann M, Nagle F, Zhou Y. The Value of Open Source Software. Harv Bus Sch [Internet]. 2024 Jan [cited 2026 Aug 28]. Available from: https://www.hbs.edu/faculty/Pages/item.aspx?num=65230

22. Software-defined vertical industries: transformation through open source [Internet]. 2020 [cited 2026 Aug 28]. Available from: https://www.linuxfoundation.org/resources/publications/software-defined-vertical-industries-transformation-through-open-source

23. Berger M. CNN [Internet]. 2001 [cited 2026 Aug 28]. IBM brings Linux to Wall Street - August 30, 2001. Available from: https://www.cnn.com/2001/TECH/industry/08/30/ibm.wallstreet.idg/index.html

24. Notes on the Generic Block Layer Rewrite in Linux 2.5 — The Linux Kernel documentation [Internet]. [cited 2026 Aug 28]. Available from: https://www.kernel.org/doc/html/v5.5/block/biodoc.html

25. The Linux Foundation. How Linux Took Over the World of Finance - Christoph H Lameter, Jump Trading LLC [Internet]. 2019 [cited 2026 Aug 28]. Available from: https://www.youtube.com/watch?v=UUOM4KdaHkY

26. LWN.net [Internet]. [cited 2026 Aug 28]. New driver API for Wireless Extensions. Available from: https://lwn.net/2001/1220/a/wireless-api.php3

27. Grokipedia [Internet]. [cited 2026 Aug 28]. Wireless tools for Linux. Available from: https://grokipedia.com/page/Wireless_tools_for_Linux

28. Chou F. Linux Wireless Networking: a short walk. Linux.com [Internet]. 2015 Mar 15 [cited 2026 Aug 28]. Available from: https://www.linux.com/training-tutorials/linux-wireless-networking-short-walk/

29. Butler S. How-To Geek [Internet]. 2026 [cited 2026 Aug 28]. How one GPL lawsuit accidentally created the homelab router that changed networking forever. Available from: https://www.howtogeek.com/a-legal-accident-turned-this-60-router-into-the-most-legendary-homelab-device-of-the-2000s/

30. OpenWrt [Internet]. [cited 2026 Aug 28]. [OpenWrt Wiki] Table of Hardware. Available from: https://openwrt.org/toh/start

31. Qualcomm Linux system software architecture - System Software Overview User Guide [Internet]. [cited 2026 Aug 28]. Available from: https://docs.qualcomm.com/doc/80-70029-252/topic/qualcomm-linux-sw-overview.html

32. Cisco [Internet]. 2021 [cited 2026 Aug 28]. Cisco NX-OS: The Advanced Network Operating System Designed for the Future. Available from: https://www.cisco.com/c/en/us/td/docs/dcn/whitepapers/nx-os-advanced-network-operating-system.html

33. Roku [Internet]. [cited 2026 Aug 28]. Roku Launchpad. Available from: https://developer.roku.com/en-ca/overview

34. Samsung [Internet]. [cited 2026 Aug 28]. Smart TV | Samsung One UI Tizen | Samsung Canada. Available from: https://www.samsung.com/ca/tvs/smart-tv/smart-hub-and-apps/

35. The Linux Foundation [Internet]. 2026 [cited 2026 Aug 28]. My Life as a Linux Kernel Developer and Maintainer with Paul E. McKenney. Available from: https://zoom.platform.linuxfoundation.org/webinar/register?project=tlf&id=7Gx04vvRIXFiYZJpqt1GptOUwZ5Y5cVU51xFqww%2BnROWllcfO8Ext3wQs9wpws0gI1OxNMTcyAZk%2BZXuaEk%3D&logo=thelinuxfoundation-color.svg&zoomID=96171875049

36. Husain A, Kampshoff P. McKinsey & Company [Internet]. 2026 [cited 2026 Sep 2]. The North American automotive industry's road to resilience. Available from: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/the-north-american-automotive-industrys-road-to-resilience

37. Automotive Grade Linux's 2025 Year in Review. Automotive Grade Linux [Internet]. 2026 Feb 19 [cited 2026 Sep 2]. Available from: https://www.automotivelinux.org/blog/automotive-grade-linuxs-2025-year-in-review/

38. Rowe R. Linux Invades Hollywood | Computer Graphics World. CGW [Internet]. 2001 Sep [cited 2026 Aug 28];24(9). Available from: https://www.cgw.com/Publications/CGW/2001/Volume-24-Issue-9-September-2001-/Linux-Invades-Hollywood.aspx

39. Robertson B. Open Source in Entertainment: How the Academy Software Foundation Creates Shared Value [Internet]. 2022 [cited 2026 Aug 28]. Available from: https://www.linuxfoundation.org/research/open-source-in-entertainment

40. Conway A. XDA [Internet]. 2026 [cited 2026 Aug 28]. Valve isn't ditching Windows or x86, but it's quietly making both optional. Available from: https://www.xda-developers.com/valve-isnt-ditching-windows-x86-quietly-making-optional/

41. Gosselin A. Linux Journal [Internet]. 2021 [cited 2026 Aug 28]. Linux in Healthcare - Cutting Costs & Adding Safety | Linux Journal. Available from: https://www.linuxjournal.com/content/linux-healthcare-cutting-costs-adding-safety

42. Stewart K, Smith J. Analyzing Open Source Interactions in Linux Based Medical Devices [ELISA] [Internet]. 2022 Jan 12 [cited 2026 Aug 28]. Available from: https://elisa.tech/blog/2022/01/12/analyzing-open-source-interactions-in-linux-based-medical-devices/

43. Muschelli J, Sweeney E, Crainiceanu CM. freesurfer: Connecting the Freesurfer software with R [Internet]. F1000Research; 2018 [cited 2026 Aug 28]. Available from: https://f1000research.com/articles/7-599 doi:10.12688/f1000research.14361.1

44. Harkey T, Baker D, Hagen J, Scott H, Palys V. Practical methods for segmentation and calculation of brain volume and intracranial volume: a guide and comparison. Quant Imaging Med Surg. 2022 Jul;12(7):3748761–3743761. doi:10.21037/qims-21-958

45. Hubbard T, Barker D, Birney E, Cameron G, Chen Y, Clark L, et al. The Ensembl genome database project. Nucleic Acids Res. 2002 Jan 1;30(1):38–41. doi:10.1093/nar/30.1.38

46. Moxa [Internet]. [cited 2026 Aug 28]. MIL Secured Edge and Built a Trusted System. Available from: https://www.moxa.com/en/spotlight/industrial-computing/moxa-industrial-linux/index

47. Toshiba [Internet]. [cited 2026 Aug 28]. (Part 1) Technical challenges of applying Linux to industrial equipment and Toshiba's longstanding efforts. Available from: https://www.global.toshiba/jp/company/digitalsolution/articles/tsoul/tech/t0901.html

48. NASA Jet Propulsion Laboratory (JPL) [Internet]. [cited 2026 Aug 28]. Meet the Open Source Software Powering Nasas Ingenuity Mars Helicopter. Available from: https://www.jpl.nasa.gov/news/meet-the-open-source-software-powering-nasas-ingenuity-mars-helicopter/

49. Projects - F Prime [Internet]. [cited 2026 Aug 28]. Available from: https://fprime.jpl.nasa.gov/overview/projects/

50. Brown E. LinuxGizmos.com [Internet]. 2021 [cited 2026 Aug 28]. NASA's Martian helicopter runs Linux. Available from: https://linuxgizmos.com/nasas-martian-helicopter-runs-linux/

51. Cohen N, Phys.org. Phys.org [Internet]. 2013 [cited 2026 Aug 28]. International Space Station making laptop migration from Windows XP to Debian 6. Available from: https://phys.org/news/2013-05-international-space-station-laptop-migration.html

52. Vaughan-Nichols S. ZDNet [Internet]. 2020 [cited 2026 Aug 28]. From Earth to orbit with Linux and SpaceX. Available from: https://www.zdnet.com/article/from-earth-to-orbit-with-linux-and-spacex/

53. Bielik J. The real story behind SpaceX's 32,000 Linux computers in orbit. Webiano Digital & Marketing Agency [Internet]. 2026 Apr 27 [cited 2026 Aug 28]. Available from: https://webiano.digital/the-real-story-behind-spacexs-32000-linux-computers-in-orbit/

54. Edge J. Introducing Space Grade Linux. LWN.net [Internet]. 2025 Sep 9 [cited 2026 Aug 28]. Available from: https://lwn.net/Articles/1036168/

55. The Linux Foundation [Internet]. [cited 2026 Aug 28]. My Life as a Linux Kernel Developer and Maintainer with Julia Lawall. Available from: https://zoom.platform.linuxfoundation.org/webinar/register?project=tlf&id=FDTYryEwaGvbVHCTY8gmh9iFqvp0eimdJC%2BXVkctFwuwYAInlUYj3MF0xU4kOZ5XXzpqqq%2BHOoW%2BVNRym9M%3D&logo=thelinuxfoundation-color.svg&zoomID=92853776811

56. Vaughan-Nichols S. ZDNET [Internet]. [cited 2026 Aug 28]. Bob Young talks about the origins of Red Hat. Available from: https://www.zdnet.com/article/bob-young-talks-about-the-origins-of-red-hat/

57. IBM Developer [Internet]. [cited 2026 Aug 28]. Get involved with open source projects - Call for Code. Available from: https://developer.ibm.com/callforcode/solutions/projects/

58. marcin.tomaszek. Linux Kernel: History, Applications, and Major Distributions | Storware BLOG. Storware [Internet]. 2022 Oct 13 [cited 2026 Aug 28]. Available from: https://storware.eu/blog/linux-kernel-history-applications-and-major-distributions/

59. Hayward D. TechRadar [Internet]. 2012 [cited 2026 Aug 28]. The history of Linux: how time has shaped the penguin. Available from: https://www.techradar.com/news/software/operating-systems/the-history-of-linux-how-time-has-shaped-the-penguin-1113914/3

60. Rootstack [Internet]. [cited 2026 Aug 28]. Red Hat: We explain everything about its cloud solutions. Available from: https://rootstack.com/en/blog/red-hat-cloud-solutions

61. Usage Statistics and Market Share of Unix for Websites, August 2026 [Internet]. [cited 2026 Aug 28]. Available from: https://w3techs.com/technologies/details/os-unix

62. Strauss D. Linux Journal [Internet]. 1998 [cited 2026 Aug 28]. Linux Helps Bring Titanic to Life | Linux Journal. Available from: https://www.linuxjournal.com/article/2494

63. MIT Media [Internet]. 2001 [cited 2026 Aug 28]. Linux takes Hollywood by storm. Available from: https://www.media.mit.edu/broadercasting/sig/archive/5-14-2001/story5.html

64. Red Hat [Internet]. [cited 2026 Aug 28]. Volkswagen supports computer-aided engineering with a modern operating system. Available from: https://www.redhat.com/en/resources/volkswagen-research-deveopment-case-study

65. TheirStack.com [Internet]. [cited 2026 Aug 28]. Companies that use Red Hat Enterprise Linux (RHEL) (8,803). Available from: https://theirstack.com/en/technology/red-hat-enterprise-linux

66. Corbet J. Clockevents and dyntick. LWN.net [Internet]. 2007 Feb 21 [cited 2026 Aug 28]. Available from: https://lwn.net/Articles/223185/

67. StatCounter Global Stats [Internet]. [cited 2026 Aug 28]. Desktop Operating System Market Share Worldwide. Available from: https://gs.statcounter.com/os-market-share/desktop/worldwide/

68. Paraschiv L. Chrome Os - everything you need to know. FOTC [Internet]. 2022 Sep 26 [cited 2026 Aug 28]. Available from: https://fotc.com/blog/chrome-os/

69. StatCounter Global Stats [Internet]. [cited 2026 Aug 28]. Mobile Operating System Market Share Worldwide. Available from: https://gs.statcounter.com/os-market-share/mobile/worldwide/

70. The 20th Anniversary of Linux. IWB Blog [Internet]. 2011 Aug 29 [cited 2026 Aug 28]. Available from: https://blog.irvingwb.com/blog/2011/08/the-20th-anniversary-of-linux.html

71. The Linux Foundation [Internet]. [cited 2026 Aug 28]. My Life as a Linux Kernel Developer and Maintainer with Shuah Khan. Available from: https://zoom.platform.linuxfoundation.org/webinar/register?project=tlf&id=kPXt5fjBXPqzGXz3OYVXe0HYE6WpzZ1rCNROoWqXxC70EYWJduoqGBVmeMNB7h6gTitSD%2BBYtcsQjkflhLk%3D&logo=thelinuxfoundation-color.svg&zoomID=99495497158

72. Susnjara S. The History of Kubernetes | IBM [Internet]. 2024 [cited 2026 Aug 28]. Available from: https://www.ibm.com/think/topics/kubernetes-history

73. Kienlt. Containers Are Just Linux Primitives: Namespaces, Cgroups, OverlayFS, and Podman. Medium [Internet]. 2026 Mar 25 [cited 2026 Aug 28]. Available from: https://medium.com/@kienlt.qn/containers-are-just-linux-primitives-namespaces-cgroups-overlayfs-and-podman-4f61d6c78fbc

74. Brown N. Adventures in a Cloud Native Landscape [Internet]. 2014 [cited 2026 Aug 28]. Introducing Namespaces. Available from: https://windsock.io/introducing-the-namespaces/

75. Corbet J. Notes from a container. LWN.net [Internet]. 2007 Oct 29 [cited 2026 Aug 28]. Available from: https://lwn.net/Articles/256389/

76. Larabel M. Phoronix [Internet]. [cited 2026 Aug 28]. OverlayFS Proposed For The Linux 3.18 Kernel. Available from: https://www.phoronix.com/news/MTc5OTc

77. torvalds. GitHub [Internet]. [cited 2026 Aug 28]. overlay filesystem · torvalds/linux@e9be9d5. Available from: https://github.com/torvalds/linux/commit/e9be9d5e76e34872f0c37d72e25bc27fe9e2c54c

78. Jaimes R. Linux Statistics 2026: Operating System Usage And Developer Adoption Rate. Quantumrun Foresight [Internet]. 2026 May 29 [cited 2026 Aug 28]. Available from: https://www.quantumrun.com/consulting/linux-statistics/

79. Red Hat Enterprise Linux Revenue Statistics [Internet]. 2025 Dec 23 [cited 2026 Aug 28]. Available from: https://commandlinux.com/statistics/red-hat-enterprise-linux-revenue-statistics/

80. The State of eBPF [Internet]. The Linux Foundation; 2024 [cited 2026 Aug 28]. Available from: https://ebpf.foundation/report-the-state-of-ebpf/

81. The Hybrid Cloud Mainframe: Opening up to cloud-native developers [Internet]. 2025 [cited 2026 Aug 28]. Available from: https://openmainframeproject.org/wp-content/uploads/sites/14/2025/10/Paper_Hybrid_Mainframe-1.pdf

82. Corbet J. The realtime preemption pull request. LWN.net [Internet]. 2024 Sep 20 [cited 2026 Sep 10]. Available from: https://lwn.net/Articles/990985/

83. Corbet J. Approaches to realtime Linux. LWN.net [Internet]. 2004 Oct 12 [cited 2026 Sep 10]. Available from: https://lwn.net/Articles/106010/

84. Kernel index [LWN.net] [Internet]. [cited 2026 Sep 10]. Available from: https://lwn.net/Kernel/Index/#Realtime

85. Corbet J. The realtime preemption end game — for real this time. LWN.net [Internet]. 2024 Sep 6 [cited 2026 Sep 10]. Available from: https://lwn.net/Articles/989212/

86. Real-Time Linux. Realtime Linux [Internet]. 2026 Sep 10 [cited 2026 Sep 10]. Available from: https://realtime-linux.org/

87. linux-kernel - [patch 1/1] MAINTAINERS: Add co-maintainers for time[rs] [Internet]. [cited 2026 Sep 10]. Available from: https://lists.openwall.net/linux-kernel/2024/03/25/4185

88. The Linux Associates [Internet]. [cited 2026 Aug 28]. The Future of Edge Computing with Linux: Powering the Next Wave of Innovation. Available from: https://thelinuxassociates.com/blog/the-future-of-edge-computing-with-linux/

89. Linux In Edge Computing Deployments Statistics 2026. Command Linux [Internet]. 2026 Mar 17 [cited 2026 Aug 28]. Available from: https://commandlinux.com/statistics/linux-in-edge-computing-deployments/

90. Whittaker G. Linux Journal [Internet]. 2024 [cited 2026 Aug 28]. Linux in the Edge Computing Ecosystem and IoT Gateway Technologies. Available from: https://www.linuxjournal.com/content/linux-edge-computing-ecosystem-and-iot-gateway-technologies

91. Ubuntu Core: The imm [Internet]. [cited 2026 Aug 28]. Ubuntu Core. Available from: https://ubuntu.com/core

92. Lavie D. Datadog [Internet]. 2023 [cited 2026 Aug 28]. A gentle introduction to XDP. Available from: https://www.datadoghq.com/blog/xdp-intro/

93. The Linux Foundation [Internet]. [cited 2026 Aug 28]. My Life as a Linux Kernel Developer and Maintainer with Mark Brown. Available from: https://zoom.platform.linuxfoundation.org/webinar/register?project=tlf&id=5wh9bf%2FuqZKOQz6ilOyhFJbFgl1AnSTZqDqZj842%2FdDGpKQaAsUyOC66%2BlRbg6LfLFyBwOx2h66rAcu9XGY%3D&logo=thelinuxfoundation-color.svg&zoomID=97172691626

94. Corbet J. Improved load balancing with machine learning. LWN.net [Internet]. 2025 Jul 1 [cited 2026 Aug 28]. Available from: https://lwn.net/Articles/1027096/

95. The Linux Foundation. Mentorship Session: Designing Custom Linux Schedulers with sched_ext [Internet]. 2025 [cited 2026 Aug 28]. Available from: https://www.youtube.com/watch?v=jsLjg9tGuVI

96. Zheng Y, Fan J, Fu Q, Yang Y, Zhang W, Quinn A. AgentCgroup: Understanding and Controlling OS Resources of AI Agents [Internet]. arXiv; 2026 [cited 2026 Aug 28]. Available from: http://arxiv.org/abs/2602.09345 doi:10.48550/arXiv.2602.09345

97. eunomia-bpf/agentcgroup [Python] [Internet]. eunomia-bpf; 2026 [cited 2026 Aug 28]. Available from: https://github.com/eunomia-bpf/agentcgroup

98. Alvarez E, Klieger B, Steele G. NVIDIA [Internet]. 2026 [cited 2026 Aug 28]. Building for the Rising Complexity of Agentic Systems with Extreme Co-Design. Available from: https://developer.nvidia.com/blog/building-for-the-rising-complexity-of-agentic-systems-with-extreme-co-design/

99. Hardware-Software Codesign - an overview [Internet]. [cited 2026 Aug 28]. Available from: https://www.sciencedirect.com/topics/engineering/hardware-software-codesign

100. Global Market Insights Inc. [Internet]. 2026 [cited 2026 Aug 28]. RISC-V Market Size & Share, Forecasts Report 2026-2035. Available from: https://www.gminsights.com/industry-analysis/risc-v-market

101. AI Coding Assistants [Internet]. [cited 2026 Aug 28]. Available from: https://docs.kernel.org/process/coding-assistants.html

102. Sharwood S. theregister [Internet]. 2026 [cited 2026 Aug 28]. Linus Torvalds to 'start being more hardnosed' about 'pointless pull requests' – some of which come from AIs. Available from: https://www.theregister.com/oses/2026/05/25/linus-torvalds-to-start-being-more-hardnosed-about-pointless-pull-requests-some-of-which-come-from-ais/5245549

103. sashiko-dev/sashiko [Rust] [Internet]. sashiko.dev; 2026 [cited 2026 Aug 28]. Available from: https://github.com/sashiko-dev/sashiko

104. Re: Linking Patchwork with Sashiko? - Linus Torvalds [Internet]. [cited 2026 Aug 28]. Available from: https://lore.kernel.org/linux-media/CAHk-=wi4zC+Ze8e+p3tMv8TtG_80KzsZ1syL9anBtmEh5Z40vg@mail.gmail.com/

105. Ahmad M. The Conversation [Internet]. 2026 [cited 2026 Aug 28]. Mythos AI is a cybersecurity threat, but it doesn't rewrite the rules of the game. Available from: https://theconversation.com/mythos-ai-is-a-cybersecurity-threat-but-it-doesnt-rewrite-the-rules-of-the-game-281268 doi:10.64628/AAI.476td45um

106. IBM Newsroom [Internet]. [cited 2026 Aug 28]. IBM and Red Hat Commit $5 Billion to Redefine the Future of Open Source in the AI Era. Available from: https://newsroom.ibm.com/2026-05-28-ibm-and-red-hat-commit-5-billion-to-redefine-the-future-of-open-source-in-the-ai-era

107. Risi C. InfoQ [Internet]. [cited 2026 Aug 28]. Linux Foundation Launches Akrites to Protect Critical Open Source Software from AI-Powered Threats. Available from: https://www.infoq.com/news/2026/07/akrites-open-source-ai-threats/

108. The Linux Foundation [Internet]. [cited 2026 Aug 28]. My Life as a Linux Kernel Developer and Maintainer with Sasha Levin. Available from: https://zoom.platform.linuxfoundation.org/webinar/register?project=tlf&id=r8dhKESMsJLWbcQBAtPD%2Blu6PLH%2BOmyUOlnuQ9a0RaeLYfaZkG7MlgK6t7IKVQtuk5FTB29GmWXf%2F5m2j0Y%3D&logo=thelinuxfoundation-color.svg&zoomID=98005374249

109. Vaughan-Nichols S. ZDNET [Internet]. [cited 2026 Aug 28]. “Rust makes coding fun again”: Why Linux is moving away from C, according to Greg Kroah-Hartman. Available from: https://www.zdnet.com/article/greg-kroah-hartman-linux-kernel-rust/

110. Rustify [Internet]. 2026 [cited 2026 Aug 28]. Rust in the Linux Kernel: What It Means in 2026. Available from: https://rustify.rs/articles/rust-in-linux-kernel-2026

111. Vaughan-Nichols SJ. Rust Integration in Linux Kernel Faces Challenges but Shows Progress. The New Stack [Internet]. 2025 Feb 19 [cited 2026 Aug 28]. Available from: https://thenewstack.io/rust-integration-in-linux-kernel-faces-challenges-but-shows-progress/

112. DomainsProject.org. Rusted Kernel [Internet]. [cited 2026 Aug 28]. Rust in the Linux Kernel — a version-by-version analysis. Available from: https://rusted-kernel.com/

113. Corbet J. The (successful) end of the kernel Rust experiment. LWN.net [Internet]. 2025 Dec 10 [cited 2026 Aug 28]. Available from: https://lwn.net/Articles/1049831/

114. ossf/best-practices-badge [Ruby] [Internet]. Open Source Security Foundation (OpenSSF); 2026 [cited 2026 Aug 28]. Available from: https://github.com/ossf/best-practices-badge

115. Wondrasek JA. The XZ Utils Backdoor CVE-2024-3094 and the Multi-Year Social Engineering Campaign Behind It. SoftwareSeni [Internet]. 2026 Feb 15 [cited 2026 Aug 28]. Available from: https://www.softwareseni.com/the-xz-utils-backdoor-cve-2024-3094-and-the-multi-year-social-engineering-campaign-behind-it/

116. Humble C. The New Stack [Internet]. [cited 2026 Aug 28]. The XZ Utils aftermath: Inside the mission to stop the next global backdoor before it starts. Available from: https://thenewstack.io/commonhaus-open-source-governance/

117. Linus Torvalds speaks on AI, More Adobe on Linux, big browser redesigns [Internet]. 2026 [cited 2026 Aug 28]. (Linux & Open Source News). Available from: https://podcast.thelinuxexp.com/@tlenewspodcast/episodes/linus-torvalds-speaks-on-ai-more-adobe-on-linux-big-browser-redesigns

118. Interview with Shuah Khan [Zoom]. 2026.

119. Corbet J. Development statistics for the 7.0 kernel. LWN.net [Internet]. 2026 Apr 13 [cited 2026 Aug 28]. Available from: https://lwn.net/Articles/1066723/

120. Corbet J. Development statistics for the 7.1 kernel. LWN.net [Internet]. 2026 Jun 15 [cited 2026 Aug 28]. Available from: https://lwn.net/Articles/1077425/

121. Vaughan-Nichols S. ZDNET [Internet]. 2026 [cited 2026 Aug 28]. Linux after Linus? The kernel community finally drafts a plan for replacing Torvalds. Available from: https://www.zdnet.com/article/linux-community-project-continuity-plan-for-replacing-linus-torvalds/

122. Perlow J. Mentorship in Open Source: Exploring the Intrinsic, Economic, and Career Value of Mentorship Programs [Internet]. 2023 [cited 2026 Aug 28]. Available from: https://www.linuxfoundation.org/research/mentorship-in-open-source

123. Mentorship [Internet]. [cited 2026 Aug 28]. Available from: https://mentorship.lfx.linuxfoundation.org/#projects_all

124. LF Events [Internet]. [cited 2026 Aug 28]. LF Live: Mentorship Series. Available from: https://events.linuxfoundation.org/lf-live-mentorship-series/

125. The Linux Foundation [Internet]. 2021 [cited 2026 Aug 28]. Interview with Shuah Khan, Kernel Maintainer & Linux Fellow - Linux Foundation. Available from: https://www.linuxfoundation.org/blog/blog/interview-with-shuah-khan-kernel-maintainer-linux-fellow

126. LFX [Internet]. [cited 2026 Aug 28]. Mentorship Dashboard. Available from: https://metabase.tools.lfx.dev/dashboard/63-mentorship-dashboard?daterange=&gender=&parent_project=&persona=mentee&program_name=&project_name=&status=&term=

127. The Linux Foundation [Internet]. 2026 [cited 2026 Aug 28]. LF Live Maintainer Series: My Life as a Linux Kernel Developer and Maintainer with Greg KH and Shuah Khan. Available from: https://www.linuxfoundation.org/webinars/lf-live-maintainer-series-my-life-as-a-linux-kernel-developer-and-maintainer-with-greg-kh-and-shuah-khan?hsLang=en

128. PIB [Internet]. 2025 [cited 2026 Aug 28]. India's UPI Revolution: Over 18 billion Transactions Every Month, A Global Leader in Fast Payments. Available from: https://static.pib.gov.in/WriteReadData/specificdocs/documents/2025/jul/doc2025720589601.pdf

THE LINUX FOUNDATION | Research

Founded in 2021, **Linux Foundation Research** explores the growing scale of open source collaboration, providing insight into emerging technology trends, best practices, and the global impact of open source projects. By leveraging project databases and networks and committing to best practices in quantitative and qualitative methodologies, Linux Foundation Research is creating the go-to library for open source insights for the benefit of organizations worldwide.

Copyright © 2026 **The Linux Foundation**

This report is licensed under the
Creative Commons Attribution-NoDerivatives 4.0 International Public License.

To reference this work, please cite as follows: Anna Hermansen, Hilary Carter, and Daniel German, "35 Years of Linux: The Open Source Engine of Global Infrastructure and Innovation," foreword by Dirk Hohndel, The Linux Foundation, October 2026.

facebook.com/TheLinuxFoundation

x.com/linuxfoundation

linkedin.com/company/TheLinuxFoundation