Click to prove you're human

KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko. Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, etc. KVM is open source software. The kernel component of KVM is included in mainline Linux, as of 2.6.20. The userspace component of KVM is included in mainline QEMU, as of 1.3. Blogs from people active in KVM-related virtualization development are syndicated at Free Unix-like operating system kernel Linux KernelTux the penguin, mascot of Linux[1]Linux kernel 6.1.0 kernel booting, under DebianOriginal author(s)Linus TorvaldsDeveloper(s)Community contributors Linus TorvaldsInitial release0.02 (5 October 1991; 33 years ago (1991-10-05))Stable release6.14.7[2] / 18 May 2025Preview release6.15-rc7[3] / 18 May 2025 Repositorygit.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git Written inC (C11 since 5.18, C89 before),[4] Assembly language, Rust,othersAvailable inEnglishLicenseGPL-2.0-only with Linux-syscall-note[5][6][7][a]Websitekernel.org The Linux kernel is a free and open source,[11]:4 Unix-like kernel that is used in many computer systems worldwide. The kernel was created by Linus Torvalds in 1991 and was soon adopted as the kernel for the GNU operating system (OS) which was created to be a free replacement for Unix. Since the late 1990s, it has been included in many operating system distributions, many of which are called Linux. One such Linux kernel operating system is Android which is used in many mobile and embedded devices. Most of the kernel code is written in C as supported by the GNU compiler collection (GCC) which has extensions beyond standard C.[11]:18[12] The code also contains assembly code for architecture-specific logic such as optimizing memory use and task execution.[11]:379–380 The kernel has a modular design such that modules can be integrated as software components – including dynamically loaded. The kernel is monolithic in an architectural sense since the entire OS kernel runs in kernel space. Linux is provided under the GNU General Public License version 2, although it contains files under other compatible licenses.[10] See also: History of Linux and Linux kernel version history Linus Torvalds at the LinuxCon Europe 2014 In 1991, Linus Torvalds was a computer science student enrolled at the University of Helsinki. During his time there, he began to develop an operating system as a side-project inspired by UNIX, for a personal computer.[13] He started with a task switcher in Intel 80386 assembly language and a terminal driver.[13] On 25 August 1991, Torvalds posted the following to comp.os.minix, a newsgroup on Usenet:[14] I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since April, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months [...]Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable [sic] (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(. On 17 September 1991, Torvalds prepared version 0.01 of Linux and put on the "ftp.funet.fi" – FTP server of the Finnish University and Research Network (FUNET). It was not even executable since its code still needed Minix to compile and test it.[15] On 5 October 1991, Torvalds announced the first "official" version of Linux, version 0.02.[16][15] [As] I mentioned a month ago, I'm working on a free version of a Minix-lookalike for AT-386 computers. It has finally reached the stage where it's even usable (though may not be depending on what you want), and I am willing to put out the sources for wider distribution. It is just version 0.02...but I've successfully run bash, gcc, gnu-make, gnu-sed, compress, etc. under it. Linux grew rapidly as many developers, including the MINIX community, contributed to the project.[citation needed] At the time, the GNU Project had completed many components for its free UNIX replacement, GNU, but its kernel, the GNU Hurd, was incomplete. The project adopted the Linux kernel for its OS.[17] Torvalds labeled the kernel with major version 0 to indicate that it was not yet intended for general use.[18] Version 0.11, released in December 1991, was the first version to be self-hosted; compiled on a computer running the Linux kernel. When Torvalds released version 0.12 in February 1992, he adopted the GNU General Public License version 2 (GPLv2) over his previous self-drafted license, which had not permitted commercial redistribution.[19] In contrast to Unix, all source files of Linux are freely available, including device drivers.[20] The initial success of Linux was driven by programmers and testers across the world. With the support of the POSIX APIs, through the libC that, whether needed, acts as an entry point to the kernel address space, Linux could run software and applications that had been developed for Unix.[21] The Linux kernel supports various hardware architectures, providing a common platform for software, including proprietary software. On 19 January 1992, the first post to the new newsgroup alt.os.linux was submitted.[22] On 31 March 1992, the newsgroup was renamed comp.os.linux.[23] The fact that Linux is a monolithic kernel rather than a microkernel was the topic of a debate between Andrew S. Tanenbaum, the creator of MINIX, and Torvalds.[24] The Tanenbaum–Torvalds debate started in 1992 on the Usenet group comp.os.minix as a general discussion about kernel architectures.[25][26] Version 0.96 released in May 1992 was the first capable of running the X Window System.[27][28] In March 1994, Linux 1.0.0 was released with 176,250 lines of code.[29] As indicated by the version number, it was the first version considered suitable for a production environment.[18] In June 1996, after release 1.3, Torvalds decided that Linux had evolved enough to warrant a new major number, and so labeled the next release as version 2.0.0.[30][31] Significant features of 2.0 included symmetric multiprocessing (SMP), support for more processors types and support for selecting specific hardware targets and for enabling architecture-specific features and optimizations.[21] The make *config family of commands of kbuild enable and configure options for building ad hoc kernel executables (vmlinux) and loadable modules.[32][33] Version 2.2, released on 20 January 1999,[34] improved locking granularity and SMP management, added m68k, PowerPC, Sparc64, Alpha, and other 64-bit platforms support.[35] Furthermore, it added new file systems including Microsoft's NTFS read-only capability.[35] In 1999, IBM published its patches to the Linux 2.2.13 code for the support of the S/390 architecture.[36] Version 2.4.0, released on 4 January 2001,[37] contained support for ISA Plug and Play, USB, and PC Cards. Linux 2.4 added support for the Pentium 4 and Itanium (the latter introduced the ia64 ISA that was jointly developed by Intel and Hewlett-Packard to supersede the older PA-RISC), and for the newer 64-bit MIPS processor.[38] Development for 2.4.x changed a bit in that more features were made available throughout the series, including support for Bluetooth, Logical Volume Manager (LVM) version 1, RAID support, InterMezzo and ext3 file systems. Version 2.6.0 was released on 17 December 2003.[39] The development for 2.6.x changed further towards including new features throughout the series. Among the changes that have been made in the 2.6 series are: integration of µClinux into the mainline kernel sources, PAE support, support for several new lines of CPUs, integration of Advanced Linux Sound Architecture (ALSA) into the mainline kernel sources, support for up to 232 users (up from 216), support for up to 229 process IDs (64-bit only, 32-bit architectures still limited to 215),[40] substantially increased the number of device types and the number of devices of each type, improved 64-bit support, support for file systems which support file sizes of up to 16 terabytes, in-kernel preemption, support for the Native POSIX Thread Library (NPTL), User-mode Linux integration into the mainline kernel sources, SELinux integration into the mainline kernel sources, InfiniBand support, and considerably more. Starting with 2.6.x releases, the kernel supported a large number of file systems; some designed for Linux, like ext3, ext4, FUSE, Btrfs,[41] and others native to other operating systems like JFS, XFS, Minix, Xenix, Irix, Solaris, System V, Windows and MS-DOS.[42] Though development had not used a version control system thus far, in 2002, Linux developers adopted BitKeeper, which was made freely available to them even though it was not free software. In 2005, because of efforts to reverse-engineer it, the company which owned the software revoked its support of the Linux community. In response, Torvalds and others wrote Git. The new system was written within weeks, and in two months the first official kernel made using it was released.[43] In 2005 the stable team was formed as a response to the lack of a kernel tree where people could work on bug fixes, and it would keep updating stable versions.[44] In February 2008 the linux-next tree was created to serve as a place where patches aimed to be merged during the next development cycle gathered.[45][46] Several subsystem maintainers also adopted the suffix -next for trees containing code which they mean to submit for inclusion in the next release cycle. As of January 2014[update], the in-development version of Linux is held in an unstable branch named linux-next.[47] The 20th anniversary of Linux was celebrated by Torvalds in July 2011 with the release of version 3.0.0.[30] As 2.6 had been the version number for 8 years, a new uname26 personality that reports 3.x as 2.6.40+x had to be added to the kernel so that old programs would work.[48] Version 3.0 was released on 22 July 2011.[49] On 30 May 2011, Torvalds announced that the big change was "NOTHING. Absolutely nothing." and asked, "...let's make sure we really make the next release not just an all new shiny number, but a good kernel too."[50] After the expected 6–7 weeks of the development process, it would be released near the 20th anniversary of Linux. On 11 December 2012, Torvalds decided to reduce kernel complexity by removing support for i386 processors—specifically by not having to emulate[51] the atomic CMPXCHG instruction introduced with the i486 to allow reliable mutexes—making the 3.7 kernel series the last one still supporting the original processor.[52][53] The same series unified support for the ARM processor.[54] The numbering change from 2.6.39 to 3.0, and from 3.19 to 4.0, involved no meaningful technical differentiation; the major version number was increased simply to avoid large minor numbers.[49][55] Stable 3.x.y kernels were released until 3.19 in February 2015. Version 3.11, released on 2 September 2013,[56] added many new features such as new O_TMPFILE flag for open(2) to reduce temporary file vulnerabilities, experimental AMD Radeon dynamic power management, low-latency network polling, and zswap (compressed swap cache).[57] In April 2015, Torvalds released kernel version 4.0.[30] By February 2015, Linux had received contributions from nearly 12,000 programmers from more than 1,200 companies, including some of the world's largest software and hardware vendors.[58] Version 4.1 of Linux, released in June 2015, contains over 19.5 million lines of code contributed by almost 14,000 programmers.[59] Linus Torvalds announced that kernel version 4.22 would instead be numbered 5.0 in March 2019, stating that "'5.0' doesn't mean anything more than that the 4.x numbers started getting big enough that I ran out of fingers and toes."[60] It featured many major additions such as support for the AMD Radeon FreeSync and NVIDIA Xavier display, fixes for F2FS, EXT4 and XFS, restored support for swap files on the Btrfs file system and continued work on the Intel Icelake Gen11 graphics and on the NXP i.MX8 SoCs.[61][62] This release was noticeably larger than the rest, Torvalds mentioning that "The overall changes for all of the 5.0 release are much bigger."[60] A total of 1,991 developers, of whom 334 were first-time collaborators, added more than 553,000 lines of code to version 5.8, breaking the record previously held by version 4.9.[63] Main article: Usage share of operating systems According to the Stack Overflow's annual Developer Survey of 2019, more than the 53% of all respondents have developed software for Linux and about 27% for Android,[64] although only about 25% develop with Linux-based operating systems.[65] Most websites run on Linux-based operating systems,[66][67] and all of the world's 500 most powerful supercomputers run on Linux.[68] Linux distributions bundle the kernel with system software (e.g., the GNU C Library, systemd, and other Unix utilities and daemons) and a wide selection of application software, but their usage share in desktops is low in comparison to other operating systems. Since Android, which runs on modified Linux kernel, accounts for the majority of mobile device operating systems,[69][70][71] and due to its rising use in embedded devices, Android is significantly responsible for rising use of Linux overall.[21] Redevelopment costs of Linux kernel The cost to redevelop version 2.6.0 of the Linux kernel in a traditional proprietary development setting has been estimated to be US$612 million (€467M, £394M) in 2004 prices using the COCOMO person-month estimation model.[72] In 2006, a study funded by the European Union put the redevelopment cost of kernel version 2.6.8 higher, at €882M ($1.14bn, £744M).[73] This topic was revisited in October 2008 by Amanda McPherson, Brian Proffitt, and Ron Hale-Evans. Using David A. Wheeler's methodology, they estimated redevelopment of the 2.6.25 kernel now costs $1.3bn (part of a total $10.8bn to redevelop Fedora 9).[74] Again, Garcia-Garcia and Alonso de Magdaleno from University of Oviedo (Spain) estimate that the value annually added to kernel was about €100M between 2005 and 2007 and €225M in 2008, it would cost also more than €1bn (about $1.4bn as of February 2010) to develop in the European Union.[75] As of 7 March 2011[update], using then-current LOC (lines of code) of a 2.6.x Linux kernel and wage numbers with David A. Wheeler's calculations it would cost approximately $3bn (about €2.2bn) to redevelop the Linux kernel as it keeps getting bigger. An updated calculation as of 26 September 2018[update], using then-current 20,088,609 LOC (lines of code) for the 4.14.14 Linux kernel and the current US national average programmer salary of $75,506 show that it would cost approximately $14,725,449,000 (£11,191,341,000) to rewrite the existing code.[76] Most who use Linux do so via a Linux distribution. Some distributions ship the vanilla or stable kernel. However, several vendors (such as Red Hat and Debian) maintain a customized source tree. These are usually updated at a slower pace than the vanilla branch, and they usually include all fixes from the relevant stable branch, but at the same time they can also add support for drivers or features which had not been released in the vanilla version the distribution vendor started basing its branch from. Graph of the sizes of Linux Kernel versions in millions of lines of code[77] The community of Linux kernel developers comprises about 5000–6000 members. According to the "2017 State of Linux Kernel Development", a study issued by the Linux Foundation, covering the commits for the releases 4.8 to 4.13, about 1500 developers were contributing from about 200–250 companies on average. The top 30 developers contributed a little more than 16% of the code. For companies, the top contributors are Intel (13.1%) and Red Hat (7.2%), Linaro (5.6%), IBM (4.1%), the second and fifth places are held by the 'none' (8.2%) and 'unknown' (4.1%) categories.[78] "Instead of a roadmap, there are technical guidelines. Instead of a central resource allocation, there are persons and companies who all have a stake in the further development of the Linux kernel, quite independently from one another: People like Linus Torvalds and I don't plan the kernel evolution. We don't sit there and think up the roadmap for the next two years, then assign resources to the various new features. That's because we don't have any resources. The resources are all owned by the various corporations who use and contribute to Linux, as well as by the various independent contributors out there. It's those people who own the resources who decide..."—Andrew Morton, 2005 IntelNoneRed HatLinaroUnknownIBMConsultantsSamsungSUSEGooglenearly 500 othercompanies Corporate affiliation of contributions to the Linux kernel, 4.8–4.13[78] Notable conflicts among Linux kernel developers: In July 2007, Con Kolivas announced that he would cease developing for the Linux kernel.[79][80] In July 2009, Alan Cox quit his role as the TTY layer maintainer after disagreement with Torvalds.[81] In December 2010, there was a discussion between Linux SCSI maintainer James Bottomley and SCST maintainer Vladislav Bolkhovitin about which SCSI target stack should be included in the Linux kernel.[82] This made some Linux users upset.[83] In June 2012, Torvalds made it very clear that he did not agree with NVIDIA releasing its drivers as closed.[84] In April 2014, Torvalds banned Kay Sievers from submitting patches to the Linux kernel for failing to deal with bugs that caused systemd to negatively interact with the kernel.[85] In October 2014, Lennart Poettering accused Torvalds of tolerating the rough discussion style on Linux kernel related mailing lists and of being a bad role model.[86] In March 2015, Christoph Hellwig filed a lawsuit against VMware for infringement of the copyright on the Linux kernel.[87] Linus Torvalds made it clear that he did not agree with this and similar initiatives by calling lawyers a festering disease.[88] In April 2021, a team from the University of Minnesota was found to be submitting "bad faith" patches to the kernel as part of its research. This resulted in the immediate reversion of all patches ever submitted by a member of the university. In addition, a warning was issued by a senior maintainer that any future patch from the university would be rejected on sight.[89][90] Prominent Linux kernel developers have been aware of the importance of avoiding conflicts between developers.[91] For a long time there was no code of conduct for kernel developers due to opposition by Torvalds.[92] However, a Linux Kernel Code of Conflict was introduced on 8 March 2015.[93] It was replaced on 16 September 2018 by a new Code of Conduct based on the Contributor Covenant. This coincided with a public apology by Torvalds and a brief break from kernel development.[94][95] On 30 November 2018, complying with the Code of Conduct, Jarkko Sakkinen of Intel sent out patches replacing instances of "fuck" appearing in source code comments with suitable versions focused on the word 'hug'.[96] Developers who feel treated unfairly can report this to the Linux Foundation Technical Advisory Board.[97] In July 2013, the maintainer of the USB 3.0 driver Sage Sharp asked Torvalds to address the abusive commentary in the kernel development community. In 2014, Sharp backed out of Linux kernel development, saying that "The focus on technical excellence, in combination with overloaded maintainers, and people with different cultural and social norms, means that Linux kernel maintainers are often blunt, rude, or brutal to get their job done".[98] At the linux.conf.au (LCA) conference in 2018, developers expressed the view that the culture of the community has gotten much better in the past few years. Daniel Vetter, the maintainer of the Intel drm/i915 graphics kernel driver, commented that the "rather violent language and discussion" in the kernel community has decreased or disappeared.[99] Laurent Pinchart asked developers for feedback on their experiences with the kernel community at the 2017 Embedded Linux Conference Europe. The issues brought up were discussed a few days later at the Maintainers Summit. Concerns over the lack of consistency in how maintainers responded to patches submitted by developers were echoed by Shuah Khan, the maintainer of the kernel self-test framework. Torvalds contended that there would never be consistency in the handling of patches because different kernel subsystems have, over time, adopted different development processes. Therefore, it was agreed upon that each kernel subsystem maintainer would document the rules for patch acceptance.[100] Linux is evolution, not intelligent design!—Linus Torvalds, 2005[101][102][103] The kernel source code, a.k.a. source tree, is managed in the Git version control system – also created by Torvalds.[104] As of 2021[update], the 5.11 release of the Linux kernel had around 30.34 million lines of code. Roughly 14% of the code is part of the "core," including architecture-specific code, kernel code, and memory management code, while 60% is drivers. Contributions are submitted as patches, in the form of text messages on the Linux kernel mailing list (LKML) (and often also on other mailing lists dedicated to particular subsystems). The patches must conform to a set of rules and to a formal language that, among other things, describes which lines of code are to be deleted and what others are to be added to the specified files. These patches can be automatically processed so that system administrators can apply them in order to make just some changes to the code or to incrementally upgrade to the next version.[105] Linux is distributed also in GNU zip (gzip) and bzip2 formats. A developer who wants to change the Linux kernel writes and tests a code change. Depending on how significant the change is and how many subsystems it modifies, the change will either be submitted as a single patch or in multiple patches of source code. In case of a single subsystem that is maintained by a single maintainer, these patches are sent as e-mails to the maintainer of the subsystem with the appropriate mailing list in Cc. The maintainer and the readers of the mailing list will review the patches and provide feedback. Once the review process has finished the subsystem maintainer accepts the patches in the relevant Git kernel tree. If the changes to the Linux kernel are bug fixes that are considered important enough, a pull request for the patches will be sent to Torvalds within a few days. Otherwise, a pull request will be sent to Torvalds during the next merge window. The merge window usually lasts two weeks and starts immediately after the release of the previous kernel version.[106] The Git kernel source tree names all developers who have contributed to the Linux kernel in the Credits directory and all subsystem maintainers are listed in Maintainers.[107] As with many large open-source software projects, developers are required to adhere to the Contributor Covenant, a code of conduct intended to address harassment of minority contributors.[108][109] Additionally, to prevent offense the use of inclusive terminology within the source code is mandated.[110] See also: Rust for Linux Linux is written in a special C programming language supported by GCC, a compiler that extends the C standard in many ways, for example using inline sections of code written in the assembly language (in GCC's "AT&T-style" syntax) of the target architecture. In September 2021, the GCC version requirement for compiling and building the Linux kernel increased from GCC 4.9 to 5.1, allowing the potential for the kernel to be moved from using C code based on the C89 standard to using code written with the C11 standard,[111] with the migration to the standard taking place in March 2022, with the release of Linux 5.18.[112] Initial support for the Rust programming language was added in Linux 6.1[113] which was released in December 2022,[114] with later kernel versions, such as Linux 6.2 and Linux 6.3, further improving the support.[115][116] Since 2002, code must adhere to the 21 rules comprising the Linux Kernel Coding Style.[117][118] As for most software, the kernel is versioned as a series of dot-separated numbers. For early versions, the version consisted of three or four dot-separated numbers called major release, minor release and revision.[11]:9 At that time, odd-numbered minor releases were for development and testing, while even numbered minor releases for production. The optional fourth digit indicated a patch level.[18] Development releases were indicated with a release candidate suffix (-rc). The current versioning conventions are different. The odd/even number implying dev/prod has been dropped, and a major version is indicated by the first two numbers together. While the time-frame is open for the development of the next major, the -rcN suffix is used to identify the n'th release candidate for the next version.[119] For example, the release of the version 4.16 was preceded by seven 4.16-rcN (from -rc1 to -rc7). Once a stable version is released, its maintenance is passed to the stable team. Updates to a stable release are identified by a three-number scheme (e.g., 4.16.1, 4.16.2, ...).[119] The kernel is usually built with the GNU toolchain. The GNU C compiler, GNU cc, part of the GNU Compiler Collection (GCC), is the default compiler for mainline Linux. Sequencing is handled by GNU make. The GNU Assembler (often called GAS or GNU as) outputs the object files from the GCC generated assembly code. Finally, the GNU Linker (GNU ld) produces a statically linked executable kernel file called vmlinux. Both as and ld are part of GNU Binary Utilities (binutils). GNU cc was for a long time the only compiler capable of correctly building Linux. In 2004, Intel claimed to have modified the kernel so that its C compiler was also capable of compiling it.[120] There was another such reported success in 2009, with a modified 2.6.22 version.[121][122] Support for the Intel compiler has been dropped in 2023.[123] Since 2010, effort has been underway to build Linux with Clang, an alternative compiler for the C language;[124] as of 12 April 2014, the official kernel could almost be compiled by Clang.[125][126] The project dedicated to this effort is named LLVMLinux after the LLVM compiler infrastructure upon which Clang is built.[127] LLVMLinux does not aim to fork either Linux or the LLVM, therefore it is a meta-project composed of patches that are eventually submitted to the upstream projects. By enabling Linux to be compiled by Clang, developers may benefit from shorter compilation times.[128] In 2017, developers completed upstreaming patches to support building the Linux kernel with Clang in the 4.15 release, having backported support for X86-64 and AArch64 to the 4.4, 4.9, and 4.14 branches of the stable kernel tree. Google's Pixel 2 shipped with the first Clang built Linux kernel,[129] though patches for Pixel (1st generation) did exist.[130] 2018 saw ChromeOS move to building kernels with Clang by default,[131] while Android made Clang[132] and LLVM's linker LLD[133] required for kernel builds in 2019. Google moved its production kernel used throughout its datacenters to being built with Clang in 2020.[134] Today, the ClangBuiltLinux group coordinates fixes to both Linux and LLVM to ensure compatibility, both composed of members from LLVMLinux and having upstreamed patches from LLVMLinux. Main articles: KGDB, Kernel panic, and Linux kernel oops Linux kernel panic output As with any software, problems with the Linux kernel can be difficult to troubleshoot. Common challenges relate to userspace vs. kernel space access, misuse of synchronization primitives, and incorrect hardware management.[11]:364 An oops is a non-fatal error in the kernel. After such an error, operations continue with suspect reliability.[135] A panic (generated by panic()) is a fatal error. After such an error, the kernel prints a message and halts the computer.[11]:371 The kernel provides for debugging by printing via printk() which stores messages in a circular buffer (overwriting older entries with newer). The syslog(2) system call provides for reading and clearing the message buffer and for setting the maximum log level of the messages to be sent to the console.[136] Kernel messages are also exported to userland through the /dev/kmsg interface.[137] The ftrace mechanism allow for debugging by tracing. It is used for monitoring and debugging Linux at runtime and it can analyze user space latencies due to kernel misbehavior.[138][139][140][141] Furthermore, ftrace allows users to trace Linux at boot-time.[142] kprobes and kretprobes can break into kernel execution (like debuggers in userspace) and collect information non-disruptively.[143] kprobes can be inserted into code at (almost) any address, while kretprobes work at function return. uprobes have similar purposes but they also have some differences in usage and implementation.[144] With KGDB Linux can be debugged in much the same way as userspace programs. KGDB requires an additional machine that runs GDB and that is connected to the target to be debugged using a serial cable or Ethernet.[145] The Linux kernel project integrates new code on a rolling basis. Standard operating procedure is that software checked into the project must work and compile without error. Each kernel subsystem is assigned a maintainer who is responsible for reviewing patches against the kernel code standards and keeping a queue of patches that can be submitted to Torvalds within a merge window that is usually several weeks. Patches are merged by Torvalds into the source code of the prior stable Linux kernel release, creating the release candidate (-rc) for the next stable release. Once the merge window is closed, only fixes to the new code in the development release are accepted. The -rc development release of the kernel goes through regression testing and once it is considered stable by Torvalds and the subsystem maintainers, a new version is released and the development process starts over again.[146] The Git tree that contains the Linux kernel source code is referred to as mainline Linux. Every stable kernel release originates from the mainline tree,[147] and is frequently published on kernel.org. Mainline Linux has only solid support for a small subset of the many devices that run Linux. Non-mainline support is provided by independent projects, such as Yocto or Linaro, but in many cases the kernel from the device vendor is needed.[148] Using a vendor kernel likely requires a board support package. Maintaining a kernel tree outside of mainline Linux has proven to be difficult.[149] Mainlining refers to the effort of adding support for a device to the mainline kernel,[150] while there was formerly only support in a fork or no support at all. This usually includes adding drivers or device tree files. When this is finished, the feature or security fix is considered mainlined.[151] The maintainer of the stable branch, Greg Kroah-Hartman, has applied the term Linux-like to downstream kernel forks by vendors that add millions of lines of code to the mainline kernel.[152] In 2019, Google stated that it wanted to use the mainline Linux kernel in Android so the number of kernel forks would be reduced.[153] The term Linux-like has also been applied to the Embeddable Linux Kernel Subset, which does not include the full mainline Linux kernel but a small modified subset of the code.[154] An iPod booting iPodLinux There are certain communities that develop kernels based on the official Linux. Some interesting bits of code from these forks that include Linux-libre, Compute Node Linux, INK, L4Linux, RTLinux, and User-Mode Linux (UML) have been merged into the mainline.[155] Some operating systems developed for mobile phones initially used heavily modified versions of Linux, including Google Android, Firefox OS, HP webOS, Nokia Maemo and Jolla Sailfish OS. In 2010, the Linux community criticised Google for effectively starting its own kernel tree:[156][157] This means that any drivers written for Android hardware platforms, can not get merged into the main kernel tree because they have dependencies on code that only lives in Google's kernel tree, causing it to fail to build in the kernel.org tree. Because of this, Google has now prevented a large chunk of hardware drivers and platform code from ever getting merged into the main kernel tree. Effectively creating a kernel branch that a number of different vendors are now relying on.[158]—Greg Kroah-Hartman, 2010 Today Android uses a customized Linux[159] where major changes are implemented in device drivers, but some changes to the core kernel code is required. Android developers also submit patches to the official Linux that finally can boot the Android operating system. For example, a Nexus 7 can boot and run the mainline Linux.[159] At a 2001 presentation at the Computer History Museum, Torvalds had this to say in response to a question about distributions of Linux using precisely the same kernel sources or not: They're not... well they are, and they're not. There is no single kernel. Every single distribution has their own changes. That's been going on since pretty much day one. I don't know if you may remember Yggdrasil was known for having quite extreme changes to the kernel and even today all of the major vendors have their own tweaks because they have some portion of the market they're interested in and quite frankly that's how it should be. Because if everybody expects one person, me, to be able to track everything that's not the point of GPL. That's not the point of having an open system. So actually the fact that a distribution decides that something is so important to them that they will add patches for even when it's not in the standard kernel, that's a really good sign for me. So that's for example how something like ReiserFS got added. And the reason why ReiserFS is the first journaling filesystem that was integrated in the standard kernel was not because I love Hans Reiser. It was because SUSE actually started shipping with ReiserFS as their standard kernel, which told me "ok." This is actually in production use. Normal People are doing this. They must know something I don't know. So in a very real sense what a lot of distribution houses do, they are part of this "let's make our own branch" and "let's make our changes to this." And because of the GPL, I can take the best portions of them.[160]—Linus Torvalds, 2001 Boot messages of a Linux kernel 2.6.25.17 The latest version and older versions are maintained separately. Most of the latest kernel releases were supervised by Torvalds.[161] The Linux kernel developer community maintains a stable kernel by applying fixes for software bugs that have been discovered during the development of the subsequent stable kernel. Therefore, www.kernel.org always lists two stable kernels. The next stable Linux kernel is released about 8 to 12 weeks later. Some releases are designated for long-term support as longterm with bug fix releases for two or more years.[162] Some projects have attempted to reduce the size of the Linux kernel. One of them is TinyLinux. In 2014, Josh Triplett started the -tiny source tree for a reduced size version.[163][164][165][166] See also: vmlinux Map of the Linux kernel Sankey diagram of Linux Kernel Source Lines of Code Even though seemingly contradictory, the Linux kernel is both monolithic and modular. The kernel is classified as a monolithic kernel architecturally since the entire OS runs in kernel space. The design is modular since it can be assembled from modules that in some cases are loaded and unloaded at runtime.[11]:338[167] It supports features once only available in closed source kernels of non-free operating systems. The rest of the article makes use of the UNIX and Unix-like operating systems convention of the manual pages. The number that follows the name of a command, interface, or other feature specifies the section (i.e. the type of the OS' component or feature) it belongs to. For example execve(2) refers to a system call, and exec(3) refers to a userspace library wrapper. The following is an overview of architectural design and of noteworthy features. Concurrent computing and (with the availability of enough CPU cores for tasks that are ready to run) even true parallel execution of many processes at once (each of them having one or more threads of execution) on SMP and NUMA architectures. Selection and configuration of hundreds of kernel features and drivers (using one of the make *config family of commands before building),[168][33][32] modification of kernel parameters before boot (usually by inserting instructions into the lines of the GRUB2 menu), and fine tuning of kernel behavior at run-time (using the sysctl(8) interface to /proc/sys/).[169][170][171] Configuration (again using the make *config commands) and run-time modifications of the policies[172] (via nice(2), setpriority(2), and the family of sched_*(2) syscalls) of the task schedulers that allow preemptive multitasking (both in user mode and, since the 2.6 series, in kernel mode[173][174]); the earliest eligible virtual deadline first scheduling (EEVDF) scheduler,[175] is the default scheduler of Linux since 2023 and it uses a red-black tree which can search, insert and delete process information (task struct) with O(log n) time complexity, where n is the number of runnable tasks.[176][177] Advanced memory management with paged virtual memory. Inter-process communications and synchronization mechanism. A virtual filesystem on top of several concrete filesystems (ext4, Btrfs, XFS, JFS, FAT32, and many more). Configurable I/O schedulers, ioctl(2)[178] syscall that manipulates the underlying device parameters of special files (it is a non standard system call, since arguments, returns, and semantics depends on the device driver in question), support for POSIX asynchronous I/O[179] (however, because they scale poorly with multithreaded applications, a family of Linux specific I/O system calls (io_*(2)[180]) had to be created for the management of asynchronous I/O contexts suitable for concurrent processing). OS-level virtualization (with Linux-VServer), paravirtualization and hardware-assisted virtualization (with KVM or Xen, and using QEMU for hardware emulation);[181][182][183][184][185][186] On the Xen hypervisor, the Linux kernel provides support to build Linux distributions (such as openSUSE Leap and many others) that work as Dom0, that are virtual machine host servers that provide the management environment for the user's virtual machines (DomU).[187] I/O Virtualization with VFIO and SR-IOV. Virtual Function I/O (VFIO) exposes direct device access to user space in a secure memory (IOMMU) protected environment. With VFIO, a VM Guest can directly access hardware devices on the VM Host Server. This technique improves performance, if compared both to Full virtualization and Paravirtualization. However, with VFIO, devices cannot be shared with multiple VM guests. Single Root I/O Virtualization (SR-IOV) combines the performance gains of VFIO and the ability to share a device with several VM Guests (but it requires special hardware that must be capable to appear to two or more VM guests as different devices).[188] Security mechanisms for discretionary and mandatory access control (SELinux, AppArmor, POSIX ACLs, and others).[189][190] Several types of layered communication protocols (including the Internet protocol suite). Asymmetric multiprocessing via the RPMsg subsystem. Most device drivers and kernel extensions run in kernel space (ring 0 in many CPU architectures), with full access to the hardware. Some exceptions run in user space; notable examples are filesystems based on FUSE/CUSE, and parts of UIO.[191][192] Furthermore, the X Window System and Wayland, the windowing system and display server protocols that most people use with Linux, do not run within the kernel. Differently, the actual interfacing with GPUs of graphics cards is an in-kernel subsystem called Direct Rendering Manager (DRM). Unlike standard monolithic kernels, device drivers are easily configured as modules, and loaded or unloaded while the system is running and can also be pre-empted under certain conditions in order to handle hardware interrupts correctly and to better support symmetric multiprocessing.[174] By choice, Linux has no stable device driver application binary interface.[193] Linux typically makes use of memory protection and virtual memory and can also handle non-uniform memory access,[194] however the project has absorbed µClinux which also makes it possible to run Linux on microcontrollers without virtual memory.[195] The hardware is represented in the file hierarchy. User applications interact with device drivers via entries in the /dev or /sys directories.[196] Process information is mapped into the /proc directory.[196] Various layers within Linux, also showing separation between the userland and kernel space User mode User applications bash, LibreOffice, GIMP, Blender, 0 A.D., Mozilla Firefox, ... System components init daemon:OpenRC, runit, systemd... System daemons:polkitd, smbd, sshd, udevd... Windowing system:X11, Wayland, SurfaceFlinger (Android) Graphics:Mesa, AMD Catalyst, ... Other libraries:GTK, Qt, EFL, SDL, SFML, FLTK, GNUstep, ... C standard library fopen, execv, malloc, memcpy, localtime, pthread_create... (up to 2000 subroutines)glibc aims to be fast, musl aims to be lightweight, uClibc targets embedded systems, bionic was written for Android, etc. All aim to be POSIX/SUS-compatible. Kernel mode Linux kernel stat, splice, dup, read, open, ioctl, write, mmap, close, exit, etc. (about 380 system calls)The Linux kernel System Call Interface (SCI), aims to be POSIX/SUS-compatible[197] Process scheduling subsystem IPC subsystem Memory management subsystem Virtual files subsystem Networking subsystem Other components: ALSA, DRI, evdev, klibc, LVM, device mapper, Linux Network Scheduler, NetfilterLinux Security Modules: SELinux, TOMOYO, AppArmor, Smack Hardware (CPU, main memory, data storage devices, etc.) Main pages: Linux kernel interfaces and Interfaces of the Linux kernel (Category) See also: System call, POSIX, and Single UNIX Specification Four interfaces are distinguished: two internal to the kernel, and two between the kernel and userspace. Linux started as a clone of UNIX, and aims toward POSIX and Single UNIX Specification compliance.[198] The kernel provides system calls and other interfaces that are Linux-specific. In order to be included in the official kernel, the code must comply with a set of licensing rules.[5][10] The Linux application binary interface (ABI) between the kernel and the user space has four degrees of stability (stable, testing, obsolete, removed);[199] The system calls are expected to never change in order to preserve compatibility for userspace programs that rely on them.[200] Loadable kernel modules (LKMs), by design, cannot rely on a stable ABI.[193] Therefore, they must always be recompiled whenever a new kernel executable is installed in a system, otherwise they will not be loaded. In-tree drivers that are configured to become an integral part of the kernel executable (vmlinux) are statically linked by the build process. There is no guarantee of stability of source-level in-kernel API[193] and, because of this, device driver code, as well as the code of any other kernel subsystem, must be kept updated with kernel evolution. Any developer who makes an API change is required to fix any code that breaks as the result of their change.[201] The set of the Linux kernel API that regards the interfaces exposed to user applications is fundamentally composed of UNIX and Linux-specific system calls.[202] A system call is an entry point into the Linux kernel.[203] For example, among the Linux-specific ones there is the family of the clone(2) system calls.[204] Most extensions must be enabled by defining the _GNU_SOURCE macro in a header file or when the user-land code is being compiled.[205] System calls can only be invoked via assembly instructions that enable the transition from unprivileged user space to privileged kernel space in ring 0. For this reason, the C standard library (libC) acts as a wrapper to most Linux system calls, by exposing C functions that, if needed,[206] transparently enter the kernel which will execute on behalf of the calling process.[202] For system calls not exposed by libC, such as the fast userspace mutex,[207] the library provides a function called syscall(2) which can be used to explicitly invoke them.[208] Pseudo filesystems (e.g., the sysfs and procfs filesystems) and special files (e.g., /dev/random, /dev/sda, /dev/tty, and many others) constitute another layer of interface to kernel data structures representing hardware or logical (software) devices.[209][210] Main article: Linux Standard BaseBecause of the differences existing between the hundreds of various implementations of the Linux OS, executable objects, even though they are compiled, assembled, and linked for running on a specific hardware architecture (that is, they use the ISA of the target hardware), often cannot run on different Linux distributions. This issue is mainly due to distribution-specific configurations and a set of patches applied to the code of the Linux kernel, differences in system libraries, services (daemons), filesystem hierarchies, and environment variables. The main standard concerning application and binary compatibility of Linux distributions is the Linux Standard Base (LSB).[211][212] However, the LSB goes beyond what concerns the Linux kernel, because it also defines the desktop specifications, the X libraries and Qt that have little to do with it.[213] The LSB version 5 is built upon several standards and drafts (POSIX, SUS, X/Open, File System Hierarchy (FHS), and others).[214] The parts of the LSB more relevant to the kernel are the General ABI (gABI),[215] especially the System V ABI[216][217] and the Executable and Linking Format (ELF),[218][219] and the Processor Specific ABI (psABI), for example the Core Specification for X86-64.[220][221] The standard ABI for how x86_64 user programs invoke system calls is to load the syscall number into the rax register, and the other parameters into rdi, rsi, rdx, r10, r8, and r9, and finally to put the syscall assembly instruction in the code.[222][223][224] At XDC2014, Alex Deucher

Alpha AXP 64-bit platform.[292][293][291] Linux runs as the main operating system on IBM's Summit; as of October 2019[update], all of the world's 500 fastest supercomputers run some operating system based on the Linux kernel,[294] a big change from 1998 when the first Linux supercomputer got added to the list.[295] Linux has also been ported to various handheld devices such as Apple's iPhone 3G and iPod.[296] In 2007, the LKDDb project has been started to build a comprehensive database of hardware and protocols known by Linux kernels.[297] The database is built automatically by static analysis of the kernel sources. Later in 2014, the Linux Hardware project was launched to automatically collect a database of all tested hardware configurations with the help of users of various Linux distributions.[298] This section needs to be updated. Please help update this article to reflect recent events or newly available information. Last update: April 2015 (September 2023) Rebootless updates can even be applied to the kernel by using live patching technologies such as Ksplice, kpatch and kGraft. Minimalistic foundations for live kernel patching were merged into the Linux kernel mainline in kernel version 4.0, which was released on 12 April 2015. Those foundations, known as livepatch and based primarily on the kernel's ftrace functionality, form a common core capable of supporting hot patching by both kGraft and kpatch, by providing an application programming interface (API) for kernel modules that contain hot patches and an application binary interface (ABI) for the userspace management utilities. However, the common core included into Linux kernel 4.0 supports only the x86 architecture and does not provide any mechanisms for ensuring function-level consistency while the hot patches are applied. Kernel bugs present potential security issues. For example, they may allow for privilege escalation or create denial-of-service attack vectors. Over the years, numerous bugs affecting system security were found and fixed.[299] New features are frequently implemented to improve the kernel's security.[300][301] Capabilities(7) have already been introduced in the section about the processes and threads. Android makes use of them and systemd gives administrators detailed control over the capabilities of processes.[302] Linux offers a wealth of mechanisms to reduce kernel attack surface and improve security which are collectively known as the Linux Security Modules (LSM).[303] They comprise the Security-Enhanced Linux (SELinux) module, whose code has been originally developed and then released to the public by the NSA,[304] and AppArmor[190] among others. SELinux is now actively developed and maintained on GitHub.[189] SELinux and AppArmor provide support to access control security policies, including mandatory access control (MAC), though they profoundly differ in complexity and scope. Another security feature is the Seccomp BPF (SECure COMPuting with Berkeley Packet Filters) which works by filtering parameters and reducing the set of system calls available to user-land applications.[305] Critics have accused kernel developers of covering up security flaws, or at least not announcing them; in 2008, Torvalds responded to this with the following:[306][307] I personally consider security bugs to be just "normal bugs". I don't cover them up, but I also don't have any reason what-so-ever to think it's a good idea to track them and announce them as something special...one reason I refuse to bother with the whole security circus is that I think it glorifies—and thus encourages—the wrong behavior. It makes "heroes" out of security people, as if the people who don't just fix normal bugs aren't as important. In fact, all the boring normal bugs are way more important, just because there's[sic] a lot more of them. I don't think some spectacular security hole should be glorified or cared about as being any more "special" than a random spectacular crash due to bad locking. Linux distributions typically release security updates to fix vulnerabilities in the Linux kernel. Many offer long-term support releases that receive security updates for a certain Linux kernel version for an extended period of time. Initially, Torvalds released Linux under a license which forbade any commercial use.[308] This was changed in version 0.12 by a switch to the GNU General Public License version 2 (GPLv2).[19] This license allows distribution and sale of possibly modified and unmodified versions of Linux but requires that all those copies be released under the same license and be accompanied by - or that, on request, free access is given to - the complete corresponding source code.[309] Torvalds has described licensing Linux under the GPLv2 as the "best thing I ever did".[308] The Linux kernel is licensed explicitly under GNU General Public License version 2 only (GPL-2.0-only) with an explicit syscall exception (Linux-syscall-note),[5][8][9] without offering the licensee the option to choose any later version, which is a common GPL extension. Contributed code must be available under GPL-compatible license.[10][201] There was considerable debate about how easily the license could be changed to use later GPL versions (including version 3), and whether this change is even desirable.[310] Torvalds himself specifically indicated upon the release of version 2.4.0 that his own code is released only under version 2.[311] However, the terms of the GPL state that if no version is specified, then any version may be used,[312] and Alan Cox pointed out that very few other Linux contributors had specified a particular version of the GPL.[313] In September 2006, a survey of 29 key kernel programmers indicated that 28 preferred GPLv2 to the then-current GPLv3 draft. Torvalds commented, "I think a number of outsiders... believed that I personally was just the odd man out because I've been so publicly not a huge fan of the GPLv3."[314] This group of high-profile kernel developers, including Torvalds, Greg Kroah-Hartman and Andrew Morton, commented on mass media about their objections to the GPLv3.[315] They referred to clauses regarding DRM/tivoization, patents, "additional restrictions" and warned a Balkanisation of the "Open Source Universe" by the GPLv3.[315][316] Torvalds, who decided not to adopt the GPLv3 for the Linux kernel, reiterated his criticism even years later.[317] It is debated whether some loadable kernel modules (LKMs) are to be considered derivative works under copyright law, and thereby whether or not they fall under the terms of the GPL. In accordance with the license rules, LKMs using only a public subset of the kernel interfaces[226][227] are non-derived works, thus Linux gives system administrators the mechanisms to load out-of-tree binary objects into the kernel address space.[10] There are some out-of-tree loadable modules that make legitimate use of the dma_buf kernel feature.[318] GPL compliant code can certainly use it. However, a different possible use case would be Nvidia Optimus that pairs a fast GPU with an Intel integrated GPU, where the Nvidia GPU writes into the Intel framebuffer when it is active. But, Nvidia cannot use this infrastructure because it necessitates bypassing a rule that can only be used by LKMs that are also GPL.[228] Alan Cox replied on LKML, rejecting a request from one of Nvidia's engineers to remove this technical enforcement from the API.[319] Torvalds clearly stated on the LKML that "[I] claim that binary-only kernel modules ARE derivative "by default"".[320] On the other hand, Torvalds has also said that "[one] gray area in particular is something like a driver that was originally written for another operating system (i.e., clearly not a derived work of Linux in origin). THAT is a gray area, and _that_ is the area where I personally believe that some modules may be considered to not be derived works simply because they weren't designed for Linux and don't depend on any special Linux behaviour".[321] Proprietary graphics drivers, in particular, are heavily discussed. Whenever proprietary modules are loaded into Linux, the kernel marks itself as being "tainted",[322] and therefore bug reports from tainted kernels will often be ignored by developers. The official kernel, that is the Linus git branch at the kernel.org repository, contains binary blobs released under the terms of the GNU GPLv2 license.[5][10] Linux can also search filesystems to locate binary blobs, proprietary firmware, drivers, or other executable modules, then it can load and link them into kernel space.[323] When it is needed (e.g., for accessing boot devices or for speed) firmware can be built-in to the kernel, this means building the firmware into vmlinux; however this is not always a viable option for technical or legal issues (e.g., it is not permitted to do this with firmware that is non-GPL compatible, although this is quite common nonetheless).[324] Linux is a registered trademark of Linus Torvalds in the United States, the European Union, and some other countries.[325][326] A legal battle over the trademark began in 1996, when William Della Croce, a lawyer who was never involved in the development of Linux, started requesting licensing fees for the use of the word Linux. After it was proven that the word was in common use long before Della Croce's claimed first use, the trademark was awarded to Torvalds.[327][328][329] In October 2024, kernel developer Greg Kroah-Hartman removed some kernel developers whose email addresses suggested a connection to Russia from their roles as maintainers.[330][331] Linus Torvalds responded that he did not support Russian aggression and would not revert the patch, insinuating that opponents of the patch were Russian trolls.[332] James Bottomley, a kernel developer, issued an apology for the handling of the situation and clarified that the action was a consequence of U.S. sanctions against Russia.[333] Linux portalFree and open-source software portal Linux – Family of Unix-like operating systems Linux kernel version history – Version history of the Linux kernel Comparison of operating systems Comparison of operating system kernels Microkernel – Kernel that provides fewer services than a traditional kernel Minix 3 – Unix-like operating system macOS – Operating system for Apple computers Microsoft Windows – Computer operating systems ^ As a whole, Linux source is provided under the terms of the GPL-2.0-only license with an explicit syscall exception.[8][9] Aside from that, individual files can be provided under a different license which is required to be compatible with the GPL-2.0-only license (i.e., the GNU General Public License version 2) or a dual license, with one of the choices being the GPL version 2 or a GPLv2 compatible license.[10] ^ "Linux Logos and Mascots". Linux Online. 2008. Archived from the original on 15 August 2010. Retrieved 19 September 2024. ^ Greg Kroah-Hartman (18 May 2025). "Linux 6.14.7". Retrieved 18 May 2025. ^ Linus Torvalds (18 May 2025). "Linux 6.15-rc7". Retrieved 18 May 2025. ^ Bergmann, Arnd (3 March 2022). "Kbuild: move to -std=gnu11". git.kernel.org. ^ a b c d "COPYING". git.kernel.org. Archived from the original on 2 September 2021. Retrieved 2 September 2021. ^ "GPL-2.0-only". spdx.org. Retrieved 2 September 2021. ^ "Linux-syscall-note". spdx.org. Retrieved 2 September 2021. ^ a b "GPL-2.0". git.kernel.org. ^ a b "Linux-syscall-note". git.kernel.org. ^ a b c d e f "Linux kernel licensing rules — The Linux Kernel documentation". www.kernel.org. Archived from the original on 7 March 2020. Retrieved 6 January 2020. ^ a b c d e f g h i j k l m n Love, Robert (2010). Linux kernel development (3rd ed.). Upper Saddle River, NJ: Addison-Wesley. ISBN 978-0-672-32946-3. OCLC 268788260. ^ "C Extensions (Using the GNU Compiler Collection (GCC))". gcc.gnu.org. Archived from the original on 20 November 2020. Retrieved 13 November 2020. ^ a b Richardson, Marjorie (1 November 1999). "Interview: Linus Torvalds". Linux Journal. Archived from the original on 14 May 2011. Retrieved 20 August 2009. ^ a b Torvalds, Linus Benedict (26 August 1991). "What would you like to see most in minix?". Newsgroup: comp.os.minix. Usenet: 1991Aug25.205708.9541@klaava.Helsinki.FI. Archived from the original on 9 May 2013. Retrieved 14 September 2016. ^ a b Welsh, Matt; Dalheimer, Matthias Kalle; Kaufman, Lar (1999). "1: Introduction to Linux". Running Linux (3rd ed.). Sebastopol, California: O'Reilly Media, Inc. p. 5. ISBN 1-56592-976-4. OCLC 50638246. ^ "Free minix-like kernel sources for 386-AT - Google Groups". groups.google.com. 5 October 1991. Archived from the original on 1 March 2021. Retrieved 19 March 2020. ^ Williams, Sam (March 2002). "Chapter 9: The GNU General Public License". Free as in Freedom: Richard Stallman's Crusade for Free Software. O'Reilly. ISBN 0-596-00287-4. Retrieved 12 November 2010. ^ a b c Christine Bresnahan & Richard Blum (2016). LPIC-2: Linux Professional Institute Certification Study Guide: Exam 201 and Exam 202. John Wiley & Sons. p. 107. ISBN 9781119150794. ^ a b Torvalds, Linus. "Release Notes for Linux v0.12". The Linux Kernel Archives. Archived from the original on 19 August 2007. Retrieved 21 February 2007. ^ Fred Hantelmann (2016). LINUX Start-up Guide: A self-contained introduction. Springer Science & Business Media. p. 1. ISBN 9783642607493. ^ a b c Fred Hantelmann (2016). LINUX Start-up Guide: A self-contained introduction. Springer Science & Business Media. p. 16. ISBN 9783642607493. ^ Summers, David W. (19 January 1992). "Troubles with Partitions". Newsgroup: alt.os.linux. Usenet: 1992Jan19.085628.18752@cseg01.uark.edu. Archived from the original on 2 June 2013. Retrieved 7 January 2007. ^ Clegg, Alan B. (31 March 1992). "It's here!". Newsgroup: comp.os.linux. Usenet: 1992Mar31.131811.19832@rock.concert.net. Archived from the original on 2 June 2013. Retrieved 7 January 2007. ^ "Appendix A: The Tanenbaum-Torvalds Debate". Open Sources: Voices from the Open Source Revolution. O'Reilly. 1999. ISBN 1-56592-582-3. Retrieved 22 November 2006. ^ Tanenbaum, Andy (29 January 1992). "LINUX is obsolete". Newsgroup: comp.os.minix. Usenet: 12595@star.cs.vu.nl. Archived from the original on 17 October 2011. Retrieved 10 May 2006. ^ Tanenbaum, Andy (12 May 2006). "Tanenbaum-Torvalds Debate: Part II". VU University Amsterdam. Archived from the original on 5 August 2015. Retrieved 6 January 2007. ^ "0.96 out next week". www.krsaborio.net. ^ "refs/tags/v0.96a - pub/scm/linux/kernel/git/nico/archive - Git at Google". kernel.googlesource.com. ^ Hayward, David (22 November 2012). "The history of Linux: how time has shaped the penguin". TechRadar. Archived from the original on 19 March 2020. Retrieved 26 March 2020. ^ a b c Christine Bresnahan & Richard Blum (2016). LPIC-2: Linux Professional Institute Certification Study Guide: Exam 201 and Exam 202. John Wiley & Sons. p. 108. ISBN 9781119150794. ^ Torvalds, Linus (9 June 1996). "Linux 2.0 really _is_ released." LKML (Mailing list). Archived from the original on 2 April 2015. Retrieved 8 March 2015. ^ a b "Kernel Build System — The Linux Kernel documentation". Kernel.org. Archived from the original on 22 July 2020. Retrieved 17 July 2020. ^ a b "Kconfig make config — The Linux Kernel documentation". Kernel.org. Archived from the original on 17 July 2020. Retrieved 13 September 2020. ^ Torvalds, Linus (20 January 1999). "2.2.0-final". LKML (Mailing list). Archived from the original on 2 April 2015. Retrieved 8 March 2015. ^ a b "The Wonderful World of Linux 2.2". 26 January 1999. Archived from the original on 6 November 2014. Retrieved 27 October 2008. ^ "Linux/390 Observations and Notes". linuxvm.org. Archived from the original on 26 February 2019. Retrieved 29 March 2020. ^ Torvalds, Linus (4 January 2001). "And oh, btw." LKML (Mailing list). Archived from the original on 26 January 2016. Retrieved 8 March 2015. ^ "The Wonderful World of Linux 2.4". Archived from the original on 17 March 2005. Retrieved 27 October 2008. ^ Torvalds, Linus (17 December 2003). "Linux 2.6.0". LKML (Mailing list). Archived from the original on 2 April 2015. Retrieved 28 February 2015. ^ "proc(5) - Linux manual page" (see /proc/sys/kernel/pid_max). Archived from the original on 7 February 2014. Retrieved 19 February 2014. ^ "btrfs Wiki". btrfs.wiki.kernel.org. Archived from the original on 25 April 2012. Retrieved 17 July 2020. ^ Fred Hantelmann (2016). LINUX Start-up Guide: A self-contained introduction. Springer Science & Business Media. pp. 1–2. ISBN 9783642607493. ^ Linux Kernel Mailing List (17 June 2005). "Linux 2.6.12". git-commits-head (Mailing list). Archived from the original on 26 January 2016. Retrieved 23 January 2008. ^ Kroah-Hartman, Greg (3 August 2006). "Adrian Bunk is now taking over the 2.6.16-stable branch". LKML (Mailing list). Archived from the original on 26 January 2016. Retrieved 21 February 2015. ^ Rothwell, Stephen (12 February 2008). "Announce: Linux-next (Or Andrew's dream :-))". LKML (Mailing list). Archived from the original on 24 November 2010. Retrieved 30 October 2010. ^ Corbet, Jonathan (21 October 2010). "linux-next and patch management process". LWN.net. Eklektix, Inc. Archived from the original on 21 June 2010. Retrieved 30 October 2010. ^ "The Linux Kernel Archives". Kernel.org. Archived from the original on 30 January 1998. Retrieved 22 January 2014. ^ "Add a personality to report 2.6.x version numbers [LWN.net]". lwn.net. Archived from the original on 16 July 2020. Retrieved 15 July 2020. ^ a b Torvalds, Linus (21 July 2011). "Linux 3.0 release". Linux kernel mailing list. Archived from the original on 18 October 2019. Retrieved 16 May 2013. ^ Torvalds, Linus (30 May 2011). "Linux 3.0-rc1". LKML (Mailing list). Archived from the original on 31 May 2011. Retrieved 1 July 2013. ^ Torvalds, Linus (10 April 2001). "Re: [PATCH] i386 rw_semaphores fix". yarchive.net. Retrieved 26 May 2024. ^ Vaughan-Nichols, Steven J. (13 December 2012). "Good-Bye 386: Linux to drop support for i386 chips with next major release". ZDNet. CBS Interactive. Archived from the original on 17 February 2015. Retrieved 6 February 2013. ^ Fingas, Jon (15 December 2012). "Linux to drop i386 support in the 3.8 kernel, make us upgrade our Doom rig". Engadget. AOL. Archived from the original on 2 April 2015. Retrieved 22 March 2015. ^ Vaughan-Nichols, Steven J. (11 December 2012). "Linux 3.7 arrives, ARM developers rejoice". ZDNet. CBS Interactive. Archived from the original on 5 November 2014. Retrieved 6 February 2013. ^ Torvalds, Linus (12 April 2015). "Linux 4.0 released". LKML (Mailing list). Archived from the original on 13 April 2015. Retrieved 12 April 2015. ^ Torvalds, Linus (2 September 2013). "Linux 3.11". LKML (Mailing list). Archived from the original on 26 February 2014. Retrieved 3 September 2013. ^ "Linux 3.11". kernelnewbies.org. 2 September 2013. Retrieved 21 January 2014. ^ "The Linux Foundation Releases Linux Development Report". Linux Foundation. 18 February 2015. Archived from the original on 19 July 2016. Retrieved 20 February 2015. ^ Michael Larabel (23 June 2014). "Linux Kernel At 19.5 Million Lines Of Code, Continues Rising". Phoronix. Archived from the original on 23 November 2020. Retrieved 23 June 2015. ^ a b Torvalds, Linus (3 March 2019). "Linus Torvalds: Linux 5.0". LKML. Archived from the original on 25 August 2024. Retrieved 25 August 2024. ^ Prakash, Abhishek (11 January 2023). "Linux Kernel 5.0 Released! Check Out The New Features". It's FOSS. Retrieved 25 August 2024. ^ Larabel, Micheal (6 January 2019). "The Many New Features & Improvements Of The Linux 5.0 Kernel". www.phoronix.com. Retrieved 25 August 2024. ^ Corbet, Jonathan (3 August 2020). "Some statistics from the 5.8 kernel cycle". LWN - Linux Weekly News. Archived from the original on 4 September 2020. Retrieved 11 August 2020. ^ "Stack Overflow Developer Survey 2019 - most popular technologies". Stack Overflow. Archived from the original on 8 October 2020. Retrieved 17 March 2020. ^ "Stack Overflow Developer Survey 2019 - development environments and tools". Stack Overflow. Archived from the original on 7 March 2020. Retrieved 17 March 2020. ^ "Usage Statistics and Market Share of Operating Systems for Websites, March 2020". w3techs.com. Retrieved 17 March 2020. ^ "Usage Statistics and Market Share of Unix for Websites, March 2020". w3techs.com. Retrieved 17 March 2020. ^ "TOP500 Supercomputer Sites: Operating system Family / Linux". Top500.org. Archived from the original on 19 November 2012. Retrieved 5 October 2019. ^ "Gartner Says Sales of Tablets Will Represent Less Than 10 Percent of All Devices in 2014" (Press release). Egham, UK: Gartner. 15 October 2014. Archived from the original on 17 October 2014. Retrieved 19 October 2014. ^ Lunden, Ingrid (15 October 2014). "Tablet Sales Growth Plummets In 2014 As Android Smartphones Continue To Soar: Gartner". TechCrunch. AOL. Archived from the original on 23 October 2014. Retrieved 23 October 2014. ^ "Global PC Shipments Exceed Forecast with Mild Improvement in Consumer Demand, While Apple Moves to #5 Spot, According to IDC" (Press release). Framingham, MA: IDC. 8 October 2014. Archived from the original on 11 October 2014. Retrieved 19 October 2014. ^ Wheeler, David A. "Linux Kernel 2.6: It's Worth More!". Archived from the original on 20 August 2011. Retrieved 18 January 2007. ^ "Economic impact of FLOSS on innovation and competitiveness of the EU ICT sector" (PDF) (Table 3 on page 50). Archived (PDF) from the original on 15 February 2010. Retrieved 8 January 2011. ^ "Estimating Total Development Cost Of a Linux Distribution" (PDF) (Table on page 6). Archived from the original (PDF) on 11 July 2010. ^ "The Billion Dollar Kernel". Linux.slashdot.org. 24 February 2010. Archived from the original on 15 May 2011. Retrieved 12 November 2010. ^ Wheeler, David. "The Linux Kernel: It's Worth More!". Archived from the original on 24 February 2021. Retrieved 17 September 2012. ^ "The Linux Kernel Archives". www.kernel.org. Retrieved 28 February 2023. ^ a b "2017 State of Linux Kernel Development". The Linux Foundation. 25 October 2017. Archived from the original on 27 May 2020. Retrieved 27 May 2020. ^ "Why I quit: kernel developer Con Kolivas". APC Magazine. ACP Magazines. 24 July 2007. Archived from the original on 7 July 2011. Retrieved 15 August 2011. ^ Corbet, Jonathan (25 July 2007). "Re: -mm merge plans for 2.6.23". LWN.net. Archived from the original on 11 February 2018. Retrieved 10 February 2018. ^ Cox, Alan (28 July 2009). "Re: [PATCH] kdesu broken". Archived from the original on 11 February 2018. Retrieved 10 February 2018. ^ Rodrigues, Goldwyn (22 January 2011). "A tale of two SCSI targets". Archived from the original on 15 February 2018. Retrieved 14 February 2018. ^ Steinmetz, Andreas (17 January 2013). "LIO - the broken iSCSI target implementation". Archived from the original on 15 February 2018. Retrieved 14 February 2018. ^ Paul, Ryan (19 June 2012). "Linus Torvalds says "f-k you" to NVIDIA". Archived from the original on 15 February 2018. Retrieved 14 February 2018. ^ John Gold (3 April 2014). "Linus Torvalds suspends key Linux developer: Kernel panic as Systemd dev pokes the bear". Archived from the original on 24 March 2019. Retrieved 24 March 2019. ^ Poettering, Lennart (6 October 2014). "On the sickness of the Linux Kernel Community". Google+. Archived from the original on 27 May 2018. Retrieved 10 February 2018. ^ Brodkin, Jon (6 March 2015). "VMware alleged to have violated Linux's open source license for years". Ars Technica. Archived from the original on 15 February 2018. Retrieved 14 February 2018. ^ McCarthy, Kieren (26 August 2016). "Having offended everyone else in the world, Linus Torvalds calls own lawyers a 'nasty festering disease'". The Register. Archived from the original on 15 February 2018. Retrieved 14 February 2018. ^ Kroah-Hartman, Greg. "[PATCH 000/190] Revertion of all of the umn.edu commits". Linux Kernel Mailing List Archive. Retrieved 13 January 2022. ^ Chin, Monica (30 April 2021). "How a university got itself banned from the Linux kernel". The Verge. Retrieved 13 January 2022. ^ Corbet, Jonathan (10 September 2007). "KS2007: Developer relations and development process". LWN.net. Archived from the original on 12 February 2018. Retrieved 11 February 2018. ^ Brodkin, Jon (16 July 2013). "Linus Torvalds defends his right to shame Linux kernel developers". ARS Technica. Archived from the original on 17 February 2018. Retrieved 11 February 2018. ^ Corbet, Jonathan (9 March 2015). "The kernel's code of conflict". LWN.net. Archived from the original on 12 February 2018. Retrieved 11 February 2018. ^ Corbet, Jonathan (18 September 2018). "Code, conflict, and conduct". LWN.net. Archived from the original on 19 September 2018. Retrieved 19 September 2018. ^ Cohen, Noam (19 September 2018). "After Years of Abusive E-mails, the Creator of Linux Steps Aside". The New Yorker. Archived from the original on 20 February 2020. Retrieved 24 September 2018. ^ Larabel, Michael. "Dropping Profanity In Kernel Code Comments: Linux Gets "Hugs"". Phoronix. Archived from the original on 21 April 2019. Retrieved 15 June 2019. ^ "Code of Conflict". Retrieved 4 February 2018.[permanent dead link] ^ Sharwood, Simon (6 October 2015). "Linux kernel dev who asked Linus Torvalds to stop verbal abuse quits over verbal abuse". The Register. Archived from the original on 29 March 2020. Retrieved 4 February 2018. ^ Edge, Jake (31 January 2018). "Too many lords, not enough stewards". LWN.net. Archived from the original on 9 November 2020. Retrieved 4 February 2018. ^ Corbet, Jonathan (6 November 2017). "Bash the kernel maintainers". LWN.net. Archived from the original on 26 January 2021. Retrieved 4 February 2018. ^ "Linux Evolution" (PDF). 26 March 2008. Archived (PDF) from the original on 14 December 2013. Retrieved 6 November 2013. ^ "Perpetual Development: A Model of the Linux Kernel Life Cycle" (PDF). 25 October 2011. Archived (PDF) from the original on 17 October 2013. Retrieved 6 November 2013. ^ Kroah-Hartman, Greg (12 February 2008). "Re: Announce: Linux-next (Or Andrew's dream :-))". Linux Kernel Mailing List (Mailing list). Archived from the original on 2 February 2017. Retrieved 30 January 2017. ^ Robert Love (2010). Linux Kernel Development: Linux Kernel Development. Pearson Education. p. 11. ISBN 9780768696790. ^ Robert Love (2010). Linux Kernel Development: Linux Kernel Development. Pearson Education. p. 12. ISBN 9780768696790. ^ "How the development process works". Archived from the original on 9 December 2017. Retrieved 4 February 2018. ^ Robert Love (2010). Linux Kernel Development: Linux Kernel Development. Pearson Education. p. 13. ISBN 9780768696790. ^ "Contributor Covenant Code of Conduct". The Linux Kernel documentation. Retrieved 6 October 2021. ^ "Linux Kernel Contributor Covenant Code of Conduct Interpretation". The Linux Kernel Documentation. Retrieved 6 October 2021. ^ Williams, Dan. "[PATCH] CodingStyle: Inclusive Terminology". Linux Kernel Mailing List. ^ Tim Anderson. "Linux kernel minimum compiler raised to GCC 5.1, allowing potential C11 use". The Register. Retrieved 12 May 2023. ^ Larabel, Michael. "The Switch Has Been Made From C89 To C11/GNU11 With Linux 5.18". www.phoronix.com. Retrieved 12 May 2023. ^ Proven, Liam. "Linux 6.1: Rust to hit mainline kernel". The Register. Retrieved 12 May 2023. ^ Sharwood, Simon (11 December 2022). "Linus Torvalds reveals Linux kernel 6.1". The Register. Retrieved 12 May 2023. ^ Corbet, Jonathan. "Rust in the 6.2 kernel". LWN.net. Retrieved 12 May 2023. ^ Larabel, Michael. "More Rust Code Readied For Linux 6.3 - Closer To Having Rust Drivers Upstreamed". Phoronix. Retrieved 12 May 2023. ^ "HOWTO do Linux kernel development — The Linux Kernel documentation". Kernel.org. Archived from the original on 7 March 2020. Retrieved 4 January 2020. ^ "Linux kernel coding style — The Linux Kernel documentation". Kernel.org. Archived from the original on 5 January 2020. Retrieved 4 January 2020. ^ a b "How the development process works — The Linux Kernel documentation". Kernel.org. Archived from the original on 9 December 2017. Retrieved 26 March 2020. ^ Kubbilun, Ingo A. (2 June 2004). "Linux kernel patch for Intel Compiler" (in German). Pyrillion.org. Archived from the original on 22 July 2011. Retrieved 12 November 2010. ^ timothy (26 February 2009). "High Performance Linux Kernel Project — LinuxDNA". Slashdot Linux. Dice Holdings. Archived from the original on 18 October 2019. Retrieved 30 October 2010. ^ Ryan, Justin (25 February 2009). "LinuxDNA Supercharges Linux with the Intel C/C++ Compiler". Linux Journal. Belltown Media, Inc. Archived from the original on 9 November 2020. Retrieved 30 October 2010. ^ Larabel, Michael (5 March 2023). "Linux 6.3 Drops Support For The Intel ICC Compiler". Phoronix. Retrieved 6 March 2023. ^ Lelbach, Bryce (25 October 2010). "Clang builds a working Linux Kernel (Boots to RL5 with SMP, networking and X, self hosts)". cfe-dev (Mailing list). Archived from the original on 7 September 2015. ^ Larabel, Michael (12 April 2014). "Linux 3.15 Can Almost Be Compiled Under LLVM's Clang". Phoronix. Archived from the original on 13 August 2020. Retrieved 10 June 2014. ^ Larabel, Michael. "Patch By Patch, LLVM Clang Gets Better At Building The Linux Kernel". Phoronix. Archived from the original on 13 August 2020. Retrieved 20 November 2014. ^ Edge, Jake (7 May 2013). "LFCS: The LLVMLinux project". LWN.net. Archived from the original on 10 August 2020. Retrieved 3 March 2015. ^ Möller, Jan-Simon (2 February 2014). "LLVMLinux: The Linux Kernel with Dragon Wings" (PDF). LLVM Project. Archived (PDF) from the original on 3 August 2020. Retrieved 3 March 2015. ^ Desaulniers, Nick; Hackmann, Greg; Hines, Stephen (18 October 2017). "2017 LLVM Developers' Meeting: Compiling Android userspace and Linux kernel with LLVM". YouTube. Archived from the original on 31 December 2020. Retrieved 7 December 2020. ^ Hackmann, Greg (2 February 2017). "marlin-nougat-mr1-clang Patch Series". Archived from the original on 10 December 2020. Retrieved 6 December 2020. ^ Kaehlcke, Matthias (22 October 2018). "cros-kernel2: Make clang the default compiler for kernel builds". Archived from the original on 10 December 2020. Retrieved 6 December 2020. ^ Larabel, Michael (4 February 2019). "Using LLVM Clang To Compile The Linux Kernel Is Heating Up Again Thanks To Google". Phoronix. Archived from the original on 25 November 2020. Retrieved 6 December 2020. ^ Desaulniers, Nick (10 December 2019). "vts: kernel: enforce vts_kernel_toolchain for all TARGET_ARCH for R". Archived from the original on 10 December 2020. Retrieved 6 December 2020. ^ Desaulniers, Nick (19 November 2020). "Re: violating function pointer signature". LKML. Retrieved 6 December 2020. ^ Bradford, John (8 March 2003). "Re: what's an OOPS". LKML (Mailing list). Archived from the original on 31 October 2014. Retrieved 30 October 2010. ^ "syslog(2) - Linux manual page". man7.org. Archived from the original on 13 October 2020. Retrieved 15 August 2020. ^ "kmsg: export printk records to the /dev/kmsg interface [LWN.net]". lwn.net. Archived from the original on 2 October 2015. Retrieved 16 August 2020. ^ See what your computer is doing with Ftrace utilities, 24 January 2019, archived from the original on 15 December 2021, retrieved 9 May 2021 ^ "Debugging the kernel using Ftrace - part 1 [LWN.net]". lwn.net. Archived from the original on 9 November 2020. Retrieved 15 September 2020. ^ "Debugging the kernel using Ftrace - part 2 [LWN.net]". lwn.net. Archived from the original on 31 March 2020. Retrieved 15 September 2020. ^ "ftrace - Function Tracer — The Linux Kernel documentation". Kernel.org. Archived from the original on 19 September 2020. Retrieved 15 September 2020. ^ "Boot-time tracing — The Linux Kernel documentation". Kernel.org. Archived from the original on 31 October 2020. Retrieved 19 September 2020. ^ "Kernel Probes (Kprobes) — The Linux Kernel documentation". Kernel.org. Archived from the original on 11 October 2020. Retrieved 6 October 2020. ^ "Uprobe-tracer: Uprobe-based Event Tracing — The Linux Kernel documentation". Kernel.org. Archived from the original on 4 December 2020. Retrieved 6 October 2020. ^ "Using kgdb, kdb and the kernel debugger internals". mirrors.edge.kernel.org. Archived from the original on 26 January 2021. Retrieved 3 November 2020. ^ Gene Sally (2010). Pro Linux Embedded Systems. Apress. p. 252. ISBN 9781430272267. ^ Billimoria, Kaiwan N. (2021). Linux Kernel Programming A Comprehensive Guide to Kernel Internals, Writing Kernel Modules, and Kernel Synchronization. Birmingham: Packt Publishing, Limited. p. 55. ISBN 978-1-78995-592-7. OCLC 1240585605. ^ Vaduva, Alexandru; Gonzalez, Alex; Simmonds, Chris (2016). Linux : embedded development : leverage the power of Linux to develop captivating and powerful embedded Linux projects : a course in three modules. Birmingham, UK: Packt Publishing. p. 663. ISBN 978-1-78712-445-5. OCLC 960471438. ^ Karim Yaghmour (2008). Building embedded Linux systems (2nd ed.). Sebastopol [Calif.]: O'Reilly Media. p. 387. ISBN 978-0-596-52968-0. OCLC 273049576. ^ Yaghmour, Karim (2011). Embedded Android. Sebastopol, CA: O'Reilly Media. p. 44. ISBN 978-1-4493-2798-9. OCLC 812180000. ^ "SoC (System on a Chip)". OpenWrt Wiki. 6 November 2014. Archived from the original on 23 August 2022. Retrieved 15 March 2021. ^ "What to do about CVE numbers [LWN.net]". lwn.net. Retrieved 15 March 2021. ^ Amadeo, Ron (20 November 2019). "Google outlines plans for mainline Linux kernel support in Android". Ars Technica. Retrieved 31 March 2021. ^ Bruchon, Jody (24 April 2021), jbruchon/elks, retrieved 27 April 2021 ^ "The state of preempt-rt". linuxplumbersconf.org. Archived from the original on 15 October 2016. Retrieved 14 June 2016. ^ Meyer, David (3 February 2010). "Linux developer explains Android kernel code removal". ZDNet. CBS Interactive. Archived from the original on 15 October 2016. Retrieved 3 February 2010. ^ "Chapter 03: maemo Platform Overview". maemo Technology Overview. Nokia. 2008. Archived from the original on 16 June 2008. Retrieved 9 April 2010. ^ Kroah-Hartman, Greg (2 February 2010). "Android and the Linux kernel community". Archived from the original on 27 April 2019. Retrieved 3 February 2010. ^ a b Roger Ye (2017). Android System Programming. Packt Publishing. p. 14. ISBN 9781787120389. ^ Torvalds, Linus (19 September 2001). "The Origins of Linux—Linus Torvalds". YouTube. Archived from the original on 15 December 2021. Retrieved 6 December 2020. ^ "Linux MAINTAINERS file". Archived from the original on 12 January 2013. ^ Alexandru Vaduva, Alex Gonzalez & Chris Simmonds (2016). Linux: Embedded Development. Packt Publishing. p. 663. ISBN 9781787124455. ^ "Linux for little systems [LWN.net]". lwn.net. Retrieved 22 November 2024. ^ "Kernel tinification [LWN.net]". lwn.net. Retrieved 22 November 2024. ^ Brown, Eric (13 April 2017). "Shrinking the Linux Kernel and File System for IoT". Linux.com. Retrieved 22 November 2024. ^ Shizukuishi, Takuya; Matsubara, Katsuya (30 March 2020). "An efficient tinification of the linux kernel for minimizing resource consumption". Proceedings of the 35th Annual ACM Symposium on Applied Computing. pp. 1228–1237. doi:10.1145/3341105.3373913. ISBN 978-1-4503-6866-7. ^ "README". git.kernel.org. Archived from the original on 24 July 2012. Retrieved 24 March 2021. ^ "KernelBuild - Linux Kernel Newbies". kernelnewbies.org. Archived from the original on 19 October 2020. Retrieved 13 September 2020. ^ "The Sysctl Interface". Linux.it. Archived from the original on 17 February 2020. Retrieved 13 September 2020. ^ "sysctl(8) - Linux manual page". man7.org. Archived from the original on 30 September 2020. Retrieved 13 September 2020. ^ "procfs(5) - Linux manual page". man7.org. Archived from the original on 24 September 2020. Retrieved 13 September 2020. ^ "sched(7) - Linux manual page". man7.org. Archived from the original on 17 July 2020. Retrieved 27 July 2020. ^ "FAQ: Preemption". kernelnewbies.org. 22 August 2009. Archived from the original on 7 August 2020. Retrieved 7 May 2015. ^ a b Jonathan Corbet (24 February 2003). "Driver porting: the preemptible kernel". LWN.net. Archived from the original on 10 August 2020. Retrieved 7 May 2015. ^ a b "EEVDF Scheduler Merged For Linux 6.6, Intel Hybrid Cluster Scheduling Re-Introduced". Phoronix. ^ a b Molnár, Ingo (13 April 2007). "[patch] Modular Scheduler Core and Completely Fair Scheduler [CFS]". LKML (Mailing list). Archived from the original on 3 November 2020. Retrieved 30 March 2020. ^ "Completely Fair Scheduler | Linux Journal". Linuxjournal.com. Archived from the original on 3 August 2020. Retrieved 30 March 2020. ^ "ioctl(2) - Linux manual page". man7.org. Archived from the original on 20 July 2020. Retrieved 11 August 2020. ^ "aio(7) - Linux manual page". man7.org. Archived from the original on 12 April 2020. Retrieved 11 August 2020. ^ "io_setup(2) - Linux manual page". man7.org. Archived from the original on 20 August 2020. Retrieved 11 August 2020. ^ "KVM". Linux-kvm.org. Archived from the original on 28 March 2020. Retrieved 29 March 2020. ^ "TechComparison - Linux Virtualization Wiki". Virt.kernelnewbies.org. Archived from the original on 3 August 2020. Retrieved 29 March 2020. ^ "Virtualization_support_through_KVM in Linux_2_6_20 - Linux Kernel Newbies". Kernelnewbies.org. Archived from the original on 29 November 2019. Retrieved 29 March 2020. ^ Coekaerts, Wim. "Linux mainline contains all the Xen code bits for Dom0 and DomU support". blogs.oracle.com. Archived from the original on 3 August 2020. Retrieved 29 March 2020. ^ "Xen celebrates full Dom0 and DomU support in Linux 3.0 – blog.xen.org". 7 June 2011. Archived from the original on 7 June 2011. Retr