WIKIPEDIA

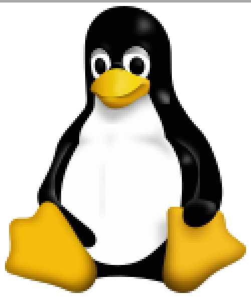
Linux kernel

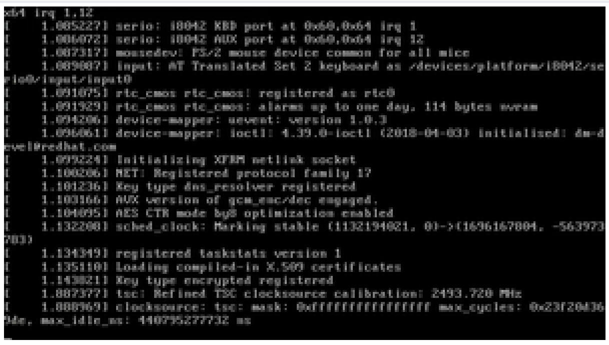
The **Linux kernel** is a free and open-source,[5][6] monolithic, modular,[7] multitasking, Unix-like operating system kernel.[8] It was conceived and created in 1991 by Linus Torvalds[9] for his i386-based PC, and it was soon adopted as the kernel for the GNU operating system,[10] which was created as a free replacement for UNIX.[11] Since then, it has spawned a large number of operating system distributions, commonly also called Linux.

Linux is deployed on a wide variety of computing systems, such as embedded devices, mobile devices (including its use in the Android operating system), personal computers, servers, mainframes, and supercomputers.[12] It can be tailored for specific architectures and for several usage scenarios using a family of simple commands (that is, without the need of manually editing its source code before compilation);[13][14][15] privileged users can also fine-tune kernel parameters at runtime.[16][17][18] Most of the Linux kernel code is written using the GNU extensions of GCC[19][20] to the standard C programming language and with the use of architecture specific instructions (ISA). This produces a highly optimized executable (vmlinux) with respect to utilization of memory space and task execution times.[21]

Day-to-day development discussions take place on the Linux kernel mailing list (LKML). Changes are tracked using the version control system git, which was created by Torvalds as a bespoke replacement for BitKeeper. Linux as a whole is released under the GNU General Public License version 2 (GPLv2),[22] but it also contains several files under other compatible licenses,[4] and an ad hoc exemption for the user space API header files (UAPI).

Linux kernel


Tux the penguin, mascot of Linux[1]



```
x64 irq 1,12
[    1.085227] serio: i8042 KBD port at 0x60,0x64 irq 1
[    1.086072] serio: i8042 AUX port at 0x60,0x64 irq 12
[    1.087317] mousedev: PS/2 mouse device common for all mice
[    1.089087] input: AT Translated Set 2 keyboard as /devices/platform/i8042/se
rio0/input/input0
[    1.091075] rtc_cmos rtc_cmos: registered as rtc0
[    1.091929] rtc_cmos rtc_cmos: alarms up to one day, 114 bytes nvram
[    1.094206] device-mapper: uevent: version 1.0.3
[    1.096061] device-mapper: ioctl: 4.39.0-ioctl (2018-04-03) initialised: dm-d
evel@redhat.com
[    1.099224] Initializing XFRM netlink socket
[    1.100206] NET: Registered protocol family 17
[    1.101236] Key type dns_resolver registered
[    1.103166] AVX version of gcm_enc/dec engaged.
[    1.104095] AES CTR mode by8 optimization enabled
[    1.132208] sched_clock: Marking stable (1132194021, 0)->(1696167804, -563973
783)
[    1.134349] registered taskstats version 1
[    1.135110] Loading compiled-in X.509 certificates
[    1.143821] Key type encrypted registered
[    1.887377] tsc: Refined TSC clocksource calibration: 2493.720 MHz
[    1.888969] clocksource: tsc: mask: 0xffffffffffffffff max_cycles: 0x23f20836
94e, max_idle_ns: 440795277732 ns
```

Linux kernel 3.0.0 booting

Developer	Linus Torvalds et al.
Written in	mostly C, with other languages including assembly
OS family	Unix-like
Initial release	0.02 (5 October 1991)
Latest release	5.13.12[2] (18 August 2021) [±] (https://en.wikipedia.org/w/index.php?title=Template:Latest_stable_software_release/Linux&action=edit)
Latest preview	5.14-rc7[3] (22 August 2021) [±] (https://en.wikipedia.org/w/index.php?title=Template:Latest_preview_software_release/Linux&action=edit)
Repository	git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git (https://git.kernel.org/pub/sc

Contents

	m/linux/kernel/git/torvalds/linux.git)
Available in	English
Kernel type	Monolithic
License	As a whole, Linux source is provided under the terms of the GNU GPLv2 (only) with an explicit syscall exception. Aside from that, individual files can be provided under a different license which is required to be compatible with the GPL-2.0 (i.e., the GPL variants) or a dual license (e.g., one of the compatible GPL variants and a permissive license like BSD, MIT etc.)[4]
Official website	www.kernel.org (https://www.kernel.org/)

History

In April 1991, Linus Torvalds, at the time a 21-year-old computer science student at the University of Helsinki, Finland, started working on some simple ideas for an operating system. He started with a task switcher in Intel 80386 assembly language and a terminal driver. On 25 August 1991, Torvalds posted the following to *comp.os.minix*, a newsgroup on Usenet:[23]

Linus Torvalds at the LinuxCon Europe 2014 in Düsseldorf

> 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 for compilation and play.[24]

On 5 October 1991, Torvalds announced the first "official" version of Linux, version 0.02.[25] At this point, Linux was able to run Bash, GCC, and some other GNU utilities:[25][24]

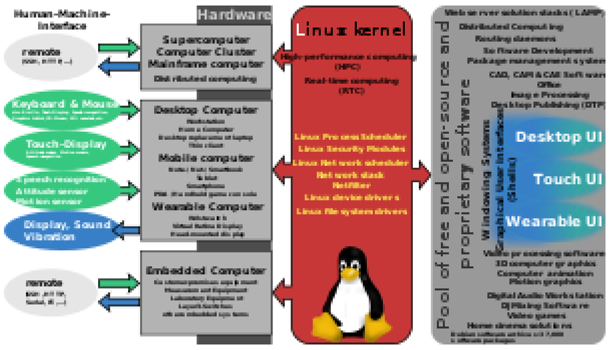
> [As] I mentioned a month ago, I'm working on a free version of a Minix-lookalike for AT-386 computers. It has finally reached the stage where it's even usable (though may not be depending on what you want), and I am willing to put out the sources for wider distribution. It is just version 0.02...but I've successfully run bash, gcc, gnu-make, gnu-sed, compress, etc. under it.

After that, many people contributed code to the project, including some developers from the MINIX community. At the time, the GNU Project had created many of the components required for a free operating system, but its own kernel, GNU Hurd, was incomplete and unavailable. The Berkeley Software Distribution had not yet freed itself from legal encumbrances. Despite the limited functionality of the early versions, Linux rapidly gained developers and users.

Torvalds assigned version 0 to the kernel to indicate that it was mainly for testing and not intended for productive use.[26] Version 0.11, released in December 1991, was the first self-hosted Linux, for it could be compiled by a computer running the same 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.[27] In contrast to Unix, all source files of Linux are freely available, including device drivers.[28] 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.[29]

On 19 January 1992, the first post to the new newsgroup *alt.os.linux* was submitted.[30] On 31 March 1992, the newsgroup was renamed *comp.os.linux*.[31] 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.[32] The Tanenbaum–Torvalds debate started in 1992 on the Usenet group *comp.os.minix* as a general discussion about kernel architectures.[33][34]



The Linux kernel supports various hardware architectures, providing a common platform for software, including proprietary software.

Linux version 0.95 was the first to be capable of running the X Window System.[35] In March 1994, Linux 1.0.0 was released with 176,250 lines of code.[36] It was the first version suitable for use in production environments.[26]

It started a versioning system for the kernel with three or four numbers separated by dots where the first represented the *major* release, the second was the *minor release*, and the third was the *revision*.[37] At that time odd-numbered *minor* releases were for development and tests, whilst even numbered *minor* releases were for production. The optional fourth digit indicated a set of patches to a *revision*.[26] Development releases were indicated with *-rc* ("release candidate") suffix.

The current version numbering is slightly different from the above. The even vs. odd numbering has been dropped and a specific *major* version is now indicated by the first two numbers, taken as a whole. 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.[38] For example, the release of the version 4.16 was preceded by seven 4.16-rcN (from -rc1 to -rc7). Once a stable release is made, its maintenance is passed off to the “stable team". Occasional updates to stable releases are identified by a three numbering scheme (e.g., 4.13.1, 4.13.2, ..., 4.13.16).[38]

After version 1.3 of the kernel, Torvalds decided that Linux had evolved enough to warrant a new *major* number, so he released version 2.0.0 in June 1996.[39][40] The series included 41 releases. The major feature of 2.0 was support for symmetric multiprocessing (SMP) and support for more types of processors.

Starting with version 2.0, Linux is configurable for selecting specific hardware targets and for enabling architecture specific features and optimizations.[29] The *make *config* family of commands of *kbuild* are used to enable and configure thousands of options for building ad hoc kernel executables (vmlinux) and loadable modules.[13][14]

Version 2.2, released on 20 January 1999,[41] improved locking granularity and SMP management, added m68k, PowerPC, Sparc64, Alpha, and other 64-bit platforms support.[42] Furthermore, it added new file systems including Microsoft's NTFS read-only capability.[42] In 1999, IBM published its patches to the Linux 2.2.13 code for the support of the S/390 architecture.[43]

Version 2.4.0, released on 4 January 2001,[44] 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.[45] Development for 2.4.*x* changed a bit in that more features were made available throughout the duration of 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.[46] The development for 2.6.*x* changed further towards including new features throughout the duration of 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 2^{32} users (up from 2^{16}), support for up to 2^{29} process IDs (64-bit only, 32-bit arches still limited to 2^{15}),[47] 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.

Also notable are the addition of a wide selection of file systems starting with the 2.6.*x* releases: now the kernel supports a large number of file systems, some that have been designed for Linux, like ext3, ext4, FUSE, Btrfs,[48] and others that are native of other operating systems like JFS, XFS, Minix, Xenix, Irix, Solaris, System V, Windows and MS-DOS.[49]

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.[50] 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.[51][52] 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, the in-development version of Linux is held in an unstable branch named *linux-next*.[53]

Linux used to be maintained without the help of an automated source code management system until, in 2002, development switched to BitKeeper. It was freely available for Linux developers but it was not free software. In 2005, because of efforts to reverse-engineer it, the company which owned the software revoked the 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.[54]

Details on the history of the 2.6 kernel series can be found in the ChangeLog files on the 2.6 kernel series source code release area of kernel.org.[55]

The 20th anniversary of Linux was celebrated by Torvalds in July 2011 with the release of the 3.0.0 kernel version.[39] As 2.6 has 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.[56]

Version 3.0 was released on 22 July 2011.[57] 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."[58] 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, making the 3.7 kernel series the last one still supporting the original processor.[59][60] The same series unified support for the ARM processor.[61]

Version 3.11, released on 2 September 2013,[62] adds many new features such as new `O_TMPFILE` flag for `open(2) (https://man7.org/linux/man-pages/man2/open.2.html)` to reduce temporary file vulnerabilities, experimental AMD Radeon dynamic power management, low-latency network polling, and zswap (compressed swap cache).[63]

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 to avoid large minor numbers.[57][64] Stable 3.x.y kernels were released until 3.19 in February 2015.

In April 2015, Torvalds released kernel version 4.0.[39] 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.[65] Version 4.1 of Linux, released in June 2015, contains over 19.5 million lines of code contributed by almost 14,000 programmers.[66]

A total of 1,991 developers, of whom 334 are first collaborators, added more than 553,000 lines of code to version 5.8, breaking the record previously held by version 4.9.[67]

According to the Stack Overflow's annual Developer Survey of 2019, more than the 53% of all respondents have developed software for Linux OS and about 27% for Android,[68] although only about 25% develop with Linux-based operating systems.[69]

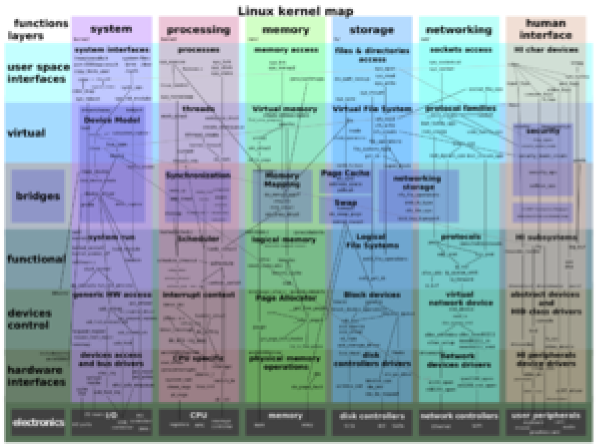
Most websites run on Linux-based operating systems,[70][71] and all of the world's 500 most powerful supercomputers use some kind of OS based on Linux.[72]

Linux distributions bundle the kernel with system software (e.g., the GNU C Library, systemd, and others 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.

Android, which accounts for the majority of the installed base of all operating systems for mobile devices,[73][74][75] is responsible for the rising usage of the Linux kernel,[29] together with its wide use in a large variety of embedded devices.

Architecture and features

Linux is a monolithic kernel with a modular design (e.g., it can insert and remove loadable kernel modules at runtime), supporting most 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 on the official manual pages. The numbers that follow the name of commands, interfaces, and other features, have the purpose of specifying the section (i.e., the type of the OS' component or feature) they belong to (e.g., `execve(2)` refers to a system call, while `exec(3)` refers to a userspace library wrapper):



Map of the Linux kernel

- 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 running compilation),[15][14][13] modification of kernel parameters before booting (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/`);[16][17][18]

- configuration (again using the `make *config` commands) and run-time modifications of the policies[76] (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[77][78]); the Completely Fair Scheduler (CFS) is the default scheduler of Linux since 2007 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;[79][80]
- 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)`[81] 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[82] (however, because they scale poorly with multithreaded applications, a family of Linux specific I/O system calls (`io_*(2)`[83]) had to be created for the management of asynchronous I/O contexts suitable for concurrently processing);
- OS-level virtualization (with Linux-VServer), paravirtualization and hardware-assisted virtualization (with KVM or Xen, and using QEMU for hardware emulation);[84][85][86][87][88][89] 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*).[90]
- security mechanisms for discretionary and mandatory access control (SELinux, AppArmor, POSIX ACLs, and others);[91][92]
- several types of layered communication protocols (including the Internet protocol suite).

Device drivers and kernel extensions run in kernel space (ring 0 in many CPU architectures), with full access to the hardware, although some exceptions run in user space, for example, filesystems based on FUSE/CUSE, and parts of UIO.[93][94] The graphics system most people use with Linux does not run within the kernel. 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.[78] By choice, Linux has no stable device driver application binary interface.[95]

Linux typically makes use of memory protection and virtual memory and can also handle non-uniform memory access,[96] however the project has absorbed μClinux which also makes it possible to run Linux on microcontrollers without virtual memory.[97]

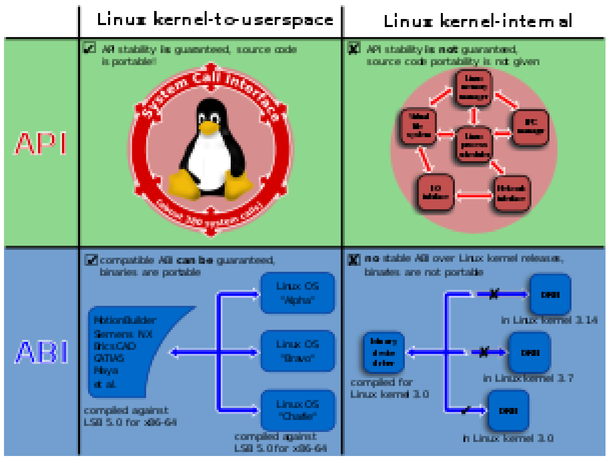
The hardware is represented in the file hierarchy. User applications interact with device drivers via entries in the `/dev` or `/sys` directories.[98] Processes information as well are mapped to the file system through the `/proc` directory.[98]

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	**Daemons**: *systemd*, *runit*, *udevd*, *polkitd*, *sshd*, *smbd*...	**Window manager**: *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* and *uClibc* target embedded systems, *bionic* 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)				
		Process scheduling subsystem	IPC subsystem	Memory management subsystem	Virtual files subsystem	Network subsystem
		Other components: ALSA, DRI, evdev, LVM, device mapper, Linux Network Scheduler, Netfilter Linux Security Modules: *SELinux*, *TOMOYO*, *AppArmor*, *Smack*				
Hardware (CPU, main memory, data storage devices, etc.)						

Interfaces

Linux is a clone of UNIX, and aims towards POSIX and Single UNIX Specification compliance.[99] The kernel also 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.[22][4]



Linux kernel-to-userspace
Linux kernel-internal
API
ABI
API stability is guaranteed, source code is portable!
System Call Interface
API stability is not guaranteed, source code portability is not given
compatible ABI can be guaranteed, binaries are portable
MotionBuilder
Siemens NX
BricsCAD
CATIAS
Maya
et al.
compiled against LSB 5.0 for x86-64
Linux OS "Alpha"
Linux OS "Bravo"
Linux OS "Charlie"
compiled against LSB 5.0 for x86-64
no stable ABI over Linux kernel releases, binaries are not portable
compiled for Linux kernel 3.0
in Linux kernel 3.14
in Linux kernel 3.7
in Linux kernel 3.0

Four interfaces are distinguished: two internal to the kernel, and two between the kernel and userspace.

The Linux Application binary interface (ABI) between the kernel and the user space has four degrees of stability (stable, testing, obsolete, removed);[100] however, the system calls are expected to never change in order to not break the userspace programs that rely on them.[101]

Loadable kernel modules (LKMs), by design, cannot rely on a stable ABI.[95] 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 building process.

There is also no guarantee of stability of source-level in-kernel API[95] and, because of this, device drivers 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.[102]

Kernel-to-userspace API

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.[103] A system call is an entry point into the Linux kernel.[104] For example, among the Linux-specific ones there is the family of the `clone(2)` system calls.[105] Most extensions must be enabled by defining the `_GNU_SOURCE` macro in a header file or when the user-land code is being compiled.[106]

System calls can only be invoked by using assembly instructions which 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, only whether it is needed,[107] can transparently enter into the kernel which will execute on behalf of the calling process.[103] For those system calls not exposed by libC, e.g. the *fast userspace mutex* (futex),[108] the library provides a function called `syscall(2)` which can be used to explicitly invoke them.[109]

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.[110][111]

Kernel-to-userspace ABI

Because 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).[112][113] 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.[114] The LSB version 5 is built upon several standards and drafts (POSIX, SUS, X/Open, File System Hierarchy (FHS), and others).[115]

The parts of the LSB largely relevant to the kernel are the *General ABI* (gABI),[116] especially the System V ABI[117][118] and the Executable and Linking Format (ELF),[119][120] and the *Processor Specific ABI* (psABI), for example the *Core Specification for X86-64*.[121][122]

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.[123][124][125]

In-kernel API

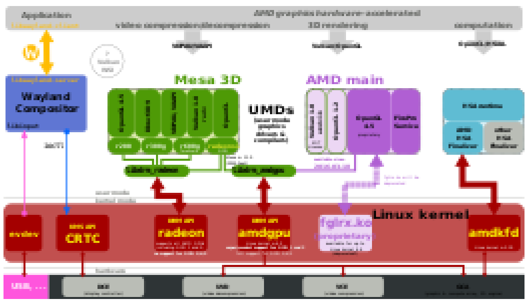
There are several kernel internal APIs utilized between the different subsystems. Some are available only within the kernel subsystems, while a somewhat limited set of in-kernel symbols (i.e., variables, data structures, and functions) is exposed also to dynamically loadable modules (e.g., device drivers loaded on demand) whether they're exported with the `EXPORT_SYMBOL()` and `EXPORT_SYMBOL_GPL()` macros[127][128] (the latter reserved to modules released under a GPL-compatible license).[129]

Linux provides in-kernel APIs that manipulate data structures (e.g., linked lists, radix trees,[130] red-black trees,[131] queues) or perform common routines (e.g., copy data from and to user space, allocate memory, print lines to the system log, and so on) that have remained stable at least since Linux version

2.6.[132][133][134]

In-kernel APIs include libraries of low-level common services used by device drivers:

- SCSI Interfaces and libATA – respectively, a peer-to-peer packet based communication protocol for storage devices attached to USB, SATA, SAS, Fibre Channel, FireWire, ATAPI device,[135] and an in-kernel library to support [S]ATA host controllers and devices.[136]
- Direct Rendering Manager (DRM) and Kernel Mode Setting (KMS) – for interfacing with GPUs and supporting the needs of modern 3D-accelerated video hardware,[137] and for setting screen resolution, color depth and refresh rate[138]
- DMA buffers (DMA-BUF) – for sharing buffers for hardware direct memory access across multiple device drivers and subsystems[139][140][141]
- Video4Linux – for video capture hardware
- Advanced Linux Sound Architecture (ALSA) – for sound cards
- New API – for network interface controllers
- mac80211 – for wireless network interface controllers[142]



At XDC2014, Alex Deucher from AMD announced the unified kernel-mode driver.[126] The proprietary Linux graphic driver, `libGL-fglrx-glx`, will share the same DRM infrastructure with Mesa 3D. As there is no stable in-kernel ABI, AMD had to constantly adapt the former binary blob used by Catalyst.

In-kernel ABI

The Linux developers choose not to maintain a stable in-kernel ABI.[143] Modules compiled for a specific version of the kernel cannot be loaded into another version without being re-compiled, assuming that the source level in-kernel API has remained the same, otherwise also the module code must be modified accordingly.[95]

Processes and threads

Linux creates processes by means of the `clone(2)` or by the newer `clone3(2)`[144] system calls. Depending on the given parameters, the new entity can share most or none of the resources of the caller. These syscalls can create new entities ranging from new independent processes (each having a special identifier called *TGID* within the *task_struct* data structure in kernel space, although that same identifier is called *PID* in userspace), to new threads of execution within the calling process (by using the `CLONE_THREAD` parameter). In this latter case the new entity owns the same *TGID* of the calling process and consequently has also the same *PID* in userspace.[145][146]

If the executable is dynamically linked to shared libraries, a dynamic linker (for ELF objects, it is typically `/lib/ld-linux.so.2`) is used to find and load the needed objects, prepare the program to run and then run it.[147]

The Native POSIX Thread Library, simply known as the NPTL,[148] provides the standard POSIX threads interface (*pthreads*) to userspace[149] Whenever a new thread is created using the pthread_create(3) POSIX interface,[150] the `clone(2)` family of system calls must also be given the address of the function that the new thread must jump to. The Linux kernel provides the `futex(7)` (acronym for "Fast user-space mutexes") mechanisms for fast user-space locking and synchronization;[151] the majority of the operations are performed in userspace but it may be necessary to communicate with the kernel using the `futex(2)` system call.[108]

A very special category of threads is the so-called *kernel threads*. They must not be confused with the above-mentioned threads of execution of the user's processes. Kernel threads exist only in kernel space and their only purpose is to concurrently run kernel tasks.[152]

Differently, whenever an independent process is created, the syscalls return exactly to the next instruction of the same program, concurrently in *parent* process and in *child's* one (i.e., one program, two processes). Different return values (one per process) enable the program to know in which of the two processes it is currently executing. Programs need this information because the child process, a few steps after process duplication, usually invokes the `execve(2)` system call (possibly via the family of `exec(3)` wrapper functions in glibC) and replace the program that is currently being run by the calling process with a new program, with newly initialized stack, heap, and (initialized and uninitialized) data segments.[153] When it is done, it results in two processes that run two different programs.

Depending on the effective user id (*euid*), and on the effective group id (*egid*), a process running with user zero privileges (*root*, the system administrator, owns the identifier 0) can perform everything (e.g., kill all the other processes or recursively wipe out whole filesystems), instead non zero user processes cannot. `capabilities(7)` divides the privileges traditionally associated with superuser into distinct units, which can be independently enabled and disabled by the parent process or dropped by the child itself.[154]

Scheduling and preemption

The Linux scheduler is modular, in the sense that it enables different scheduling classes and policies.[155][156] Scheduler classes are plugable scheduler algorithms that can be registered with the base scheduler code. Each class schedules different types of processes. The core code of the scheduler iterates over each class in order of priority and chooses the highest priority scheduler that has a schedulable entity of type struct sched_entity ready to run.[157] Entities may be threads, group of threads, and even all the processes of a specific user.

Linux provides both *user preemption* as well as full *kernel preemption*.[158] Preemption reduces latency, increases responsiveness,[159] and makes Linux more suitable for desktop and real-time applications.

For normal tasks, by default, the kernel uses the Completely Fair Scheduler (CFS) class, introduced in the 2.6.23 version of the kernel.[79] Internally this default-scheduler class is defined in a macro of a C header as `SCHED_NORMAL`. In other POSIX kernels, a similar policy known as `SCHED_OTHER` allocates CPU timeslices (i.e, it assigns absolute slices of the processor time depending on either predetermined or dynamically computed priority of each process). The Linux CFS does away with absolute timeslices and assigns a fair proportion of CPU time, as a function of parameters like the total number of runnable processes and the time they have already run; this function also takes into account a kind of weight that depends on their relative priorities (nice values).[160]

With user preemption, the kernel scheduler can replace the current process with the execution of a context switch to a different one that therefore acquires the computing resources for running (CPU, memory, and more). It makes it according to the CFS algorithm (in particular, it uses a variable called `vruntime` for sorting entities and then chooses the one that has the smaller vruntime, - i.e., the schedulable entity that has had the least share of CPU time), to the active scheduler policy and to the relative priorities.[161] With kernel preemption, the kernel can preempt itself when an interrupt handler returns, when kernel tasks block, and whenever a subsystem explicitly calls the schedule() function.

The kernel also contains two POSIX-compliant[162] real-time scheduling classes named `SCHED_FIFO` (realtime first-in-first-out) and `SCHED_RR` (realtime round-robin), both of which take precedence over the default class.[155] An additional scheduling policy known as `SCHED_DEADLINE`, implementing the earliest deadline first algorithm (EDF), was added in kernel version 3.14, released on 30 March 2014.[163][164] `SCHED_DEADLINE` takes precedence over all the other scheduling classes.

The Linux kernel patch `PREEMPT_RT` enables full preemption of critical sections, interrupt handlers, and "interrupt disable" code sequences.[165] Partial integration of the real-time Linux patches brought the above mentioned functionality to the kernel mainline.[166]

Concurrency and synchronization

The kernel has different causes of concurrency (e.g., interrupts, bottom halves, preemption of kernel and users tasks, symmetrical multiprocessing).[167] For protecting critical regions (sections of code that must be executed atomically), shared memory locations (like global variables and other data structures with global scope), and regions of memory that are asynchronously modifiable by hardware (e.g., having the C `volatile` type qualifier), Linux provides a large set of tools. They consist of atomic types (which can only be manipulated by a set of specific operators), spinlocks, semaphores, mutexes,[168][169] and lockless algorithms (e.g., RCUs).[170][171][172] Most lock-less algorithms are built on top of memory barriers for the purpose of enforcing memory ordering and prevent undesired side effects due to compiler's optimizations.[173][174][175][176]

Interrupts management

The management of the interrupts, although it could be seen as a single job, is divided in two separate parts. This split in two is due to the different time constraints and to the synchronization needs of the tasks whose the management is composed of. The first part is made up of an asyncronous interrupt service routine that in Linux is known as the *top half*, while the second part is carried out by one of three types of the so-called *bottom halves* (*softirq*, *tasklets,* and *work queues*).[177] Linux interrupts service routines can be nested (i.e., a new IRQ can trap into a high priority ISR that preempts any other lower priority ISRs).

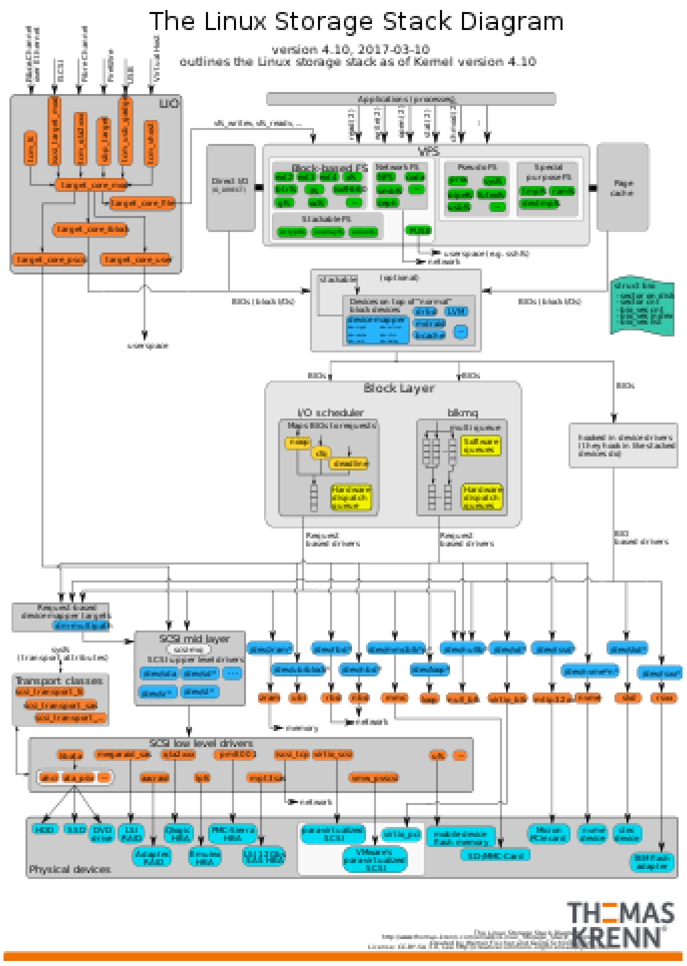
Memory management

Memory management in Linux is a complex topic. First of all, the kernel is not pageable (i.e., it is always resident in physical memory and cannot be swapped to the disk). In the kernel there is no memory protection (no *SIGSEGV* signals, unlike in userspace), therefore memory violations lead to instability and system crashes.[178]

Linux implements virtual memory with 4 and 5-levels page tables. As said, only user memory space is always pageable. It maintains information about each page frame of RAM in apposite data structures (of type `struct page`) that are populated immediately after boots and that are kept until shutdown, regardless of them being or not associated with virtual pages. Furthermore, it classifies all page frames in zones, according to their architecture dependent constraints and intended use. For example, pages reserved for DMA operations are in ZONE_DMA, pages that are not permanently mapped to virtual addresses are in ZONE_HIGHMEM (in x86_32 architecture this zone is for physical addresses above 896 MB, while x86_64 does not need it because x86_64 can permanently map physical pages that reside in higher addresses), and all that remains (with the exception of other less used classifications) is in ZONE_NORMAL.

Small chunks of memory can be dynamically allocated via the family of `kmalloc()` API and freed with the appropriate variant of `kfree()`. `vmalloc()` and `kvfree()` are used for large virtually contiguous chunks. alloc_pages() allocates the desired number of entire pages.

Supported architectures



The Linux Strage Stack Diagram[179]

While not originally designed to be portable,[23][180] Linux is now one of the most widely ported operating system kernels, running on a diverse range of systems from the ARM architecture to IBM z/Architecture mainframe computers. The first port was performed on the Motorola 68000 platform. The modifications to the kernel were so fundamental that Torvalds viewed the Motorola version as a fork and a "Linux-like operating system".[180] However, that moved Torvalds to lead a major restructure of the code to facilitate porting to more computing architectures. The first Linux that, in a single source tree, had code for more than i386 alone, supported the DEC Alpha AXP 64-bit platform.[181][182][180]

Linux runs as the main operating system on IBM's Summit; as of October 2019, all of the world's 500 fastest supercomputers run some operating system based on the Linux kernel,[12] a big change from 1998 when the first Linux supercomputer got added to the list.[183]

Linux has also been ported to various handheld devices such as Apple's iPhone 3G and iPod.[184]

Supported devices

TiVo DVR, a consumer device running Linux

In 2007, the LKDDb project has been started to build a comprehensive database of hardware and protocols known by Linux kernels.[185] 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.[186]

Live patching

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. As of April 2015, there is ongoing work on porting kpatch and kGraft to the common live patching core provided by the Linux kernel mainline.[187][188][189]

Security

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.[190] New features are frequently implemented to improve the kernel's security.[191][192]

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.[193]

Linux offers a wealth of mechanisms to reduce kernel attack surface and improve security which are collectively known as the Linux Security Modules (LSM).[194] They comprise the Security-Enhanced Linux (SELinux) module, whose code has been originally developed and then released to the public by the NSA,[195] and AppArmor[92] among others. SELinux is now actively developed and maintained on GitHub.[91] 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.[196]

Critics have accused kernel developers of covering up security flaws, or at least not announcing them; in 2008, Linus Torvalds responded to this with the following:[197][198]

> 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.

Development

Developer community

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. As of 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.[199]

> 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

Source code management

The Linux development community uses Git to manage the source code. Git users clone the latest version of Torvalds' tree with `git-clone(1)`[200] and keep it up to date using `git-pull(1)`.[201][202] Contributions are submitted as patches, in the form of text messages on the 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.[203] Linux is distributed also in GNU zip (gzip) and bzip2 formats.

Submitting code to the kernel

A developer who wants to change the Linux kernel starts with developing and testing that 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.[204] 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*.[205]

Programming language and coding style

Linux is written in a special C programming language supported by GCC, a compiler that extends in many ways the C standard, for example using inline sections of code written in the assembly language (in GCC's "AT&T-style" syntax) of the target architecture. Since 2002 all the code must adhere to the 21 rules comprising the *Linux Kernel Coding Style*.[206][207]

GNU toolchain

The GNU Compiler Collection (GCC or GNU cc) is the default compiler for the mainline Linux sources and it is invoked by a utility called make. Then, the GNU Assembler (more often called GAS or GNU as) outputs the object files from the GCC generated assembly code. Finally, the GNU Linker (GNU ld) is used to produce a statically linked executable kernel file called `vmlinux`. Both `as` and `ld` are part of GNU Binary Utilities (binutils). The above-mentioned tools are collectively known as the GNU toolchain.

Compiler compatibility

GCC 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.[208] There was another such reported success in 2009, with a modified 2.6.22 version.[209][210]

Since 2010, effort has been underway to build Linux with Clang, an alternative compiler for the C language;[211] as of 12 April 2014, the official kernel could almost be compiled by Clang.[212][213] The project dedicated to this effort is named *LLVMLinux* after the LLVM compiler infrastructure upon which Clang is built.[214] 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.[215]

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,[216] though patches for Pixel (1st generation) did exist.[217] 2018 saw ChromeOS move to building kernels with Clang by default,[218] while Android (operating system) made Clang[219] and LLVM's linker LLD[220] required for kernel builds in 2019. Google moved its production kernel used throughout its datacenters to being built with Clang in 2020.[221] Today, the *ClangBuiltLinux (https://clangbuiltlinux.github.io/)* group coordinates fixes to both Linux and LLVM to ensure compatibility, both composed of members from *LLVMLinux* and having upstreamed patches from *LLVMLinux*.

Kernel debugging

An example of Linux kernel panic

Bugs involving the Linux Kernel can be difficult to troubleshoot, this is because of the kernel's interaction with userspace and hardware; and also because they might be caused from a wider range of reasons compared to those of user programs. A few examples of the underlying causes are semantic errors in code, misuse of synchronization primitives, and incorrect hardware management.[222]

A report of a non-fatal bug in the kernel is called an "oops"; such deviations from correct behavior of the Linux kernel may allow continued operation with compromised reliability.[223]

A critical and fatal error is reported via the `panic()` function. It prints a message and then halts the kernel.[224]

One of the most common techniques used to find out bugs in code is *debugging by printing*. For this purpose Linux provides an in-kernel API called `printk()` which stores messages in a circular buffer. The `syslog(2)` system call is used for reading and/or clearing the kernel message ring buffer and for setting the maximum *log level* of the messages to be sent to the console (i.e., one of the eight `KERN_*` parameters of `printk()`, which tell the severity of the condition reported); usually it is invoked via the glibC wrapper `klogctl(3)`.[225] Kernel messages are also exported to userland through the */dev/kmsg* interface[226] (e.g., systemd-journald[227][228] reads that interface and by default append the messages to `/var/log/journal`).

Another fundamental technique for debugging a running kernel is tracing. The *ftrace* mechanism is a Linux internal tracer; it is used for monitoring and debugging Linux at runtime and it can also analyze user space latencies due to kernel misbehavior.[229][230][231][232] Furthermore, *ftrace* allows users to trace Linux at boot-time.[233]

kprobes and *kretprobes* can break (like debuggers in userspace) into Linux and non-disruptively collect information.[234] *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.[235]

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.[236]

Development model

The Linux kernel project integrates new code on a rolling basis. 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 keeps a queue of patches that can be submitted to Linus Torvalds within a merge window of several weeks. Patches are merged by Torvalds into the source code of the prior stable Linux kernel release, creating the *-rc* release candidate for the next stable kernel. 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 tests and once it is judged to be stable by Torvalds and the kernel subsystem maintainers a new Linux kernel is released and the development process starts all over again.[237]

Developers who feel treated unfairly can report this to the Linux Foundation's Technical Advisory Board.[238] In July 2013, the maintainer of the USB 3.0 driver Sarah 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".[239] 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.[240]

Laurent Pinchart asked developers for feedback on their experience 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.[241]

Mainline Linux

The Git tree of Linus Torvalds that contains the Linux kernel is referred to as **mainline Linux**. Every stable kernel release originates from the mainline tree,[242] 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.[243] Using a vendor kernel likely requires a board support package.

Maintaining a kernel tree outside of mainline Linux has proven to be difficult.[244]

Mainlining refers to the effort of adding support for a device to the mainline kernel,[245] 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*.[246]

Linux-like kernel

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.[247] In 2019, Google stated that they wanted to use the mainline Linux kernel in Android so the number of kernel forks would be reduced.[248] 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.[249]

Linux forks

There are certain communities that develop kernels based on the official Linux. Some interesting bits of code from these *forks* (i.e., a slang term meaning "derived projects") that include Linux-libre, Compute Node Linux, INK, L4Linux, RTLinux, and User-Mode Linux (UML) have been merged into the mainline.[250] 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:[251][252]

An iPod booting iPodLinux

> 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.[253]
>
> — Greg Kroah-Hartman, 2010

Today Android uses a slightly customized Linux[254] where changes are implemented in device drivers so that little or no change 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.[254]

At a 2001 presentation at the Computer History Museum, Linus 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.*[255]
>
> — Linus Torvalds, 2001

Development community conflicts

There have been several notable conflicts among Linux kernel developers. Examples of such conflicts are:

- In July 2007, Con Kolivas announced that he would cease developing for the Linux kernel.[256][257]
- In July 2009, Alan Cox quit his role as the TTY layer maintainer after disagreement with Linus Torvalds.[258]
- 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.[259] This made some Linux users upset.[260]
- In June 2012, Torvalds made it very clear that he did not agree with NVIDIA releasing its drivers as closed.[261]
- 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.[262]
- 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.[263]
- In March 2015, Christoph Hellwig filed a lawsuit against VMware for infringement of the copyright on the Linux kernel.[264] Linus Torvalds made it clear that he did not agree with this and similar initiatives by calling lawyers a festering disease.[265]

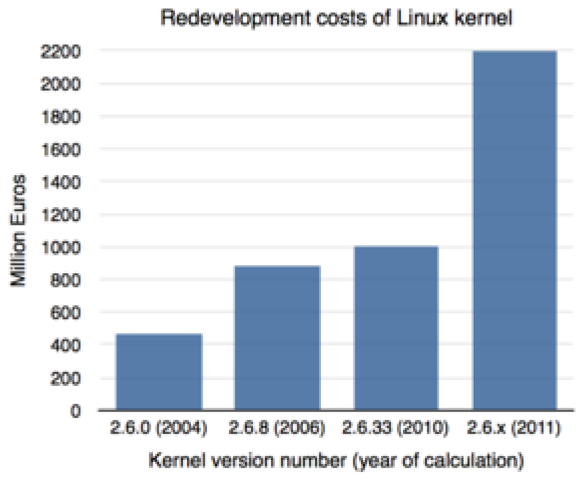
Prominent Linux kernel developers have been aware of the importance of avoiding conflicts between developers.[266] For a long time there was no code of conduct for kernel developers due to opposition by Linus Torvalds.[267] However, a Linux Kernel *Code of Conflict* was introduced on 8 March 2015.[268] 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.[269][270] 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'.[271]

Codebase

As of 2021, 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" (arch, kernel and mm directories) while 60% is drivers.

> Linux is evolution, not intelligent design!
>
> — Linus Torvalds, 2005[272][273][274]

Estimated cost to redevelop



Redevelopment costs of Linux kernel

The cost to redevelop the Linux kernel version 2.6.0 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.[275] 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).[276]

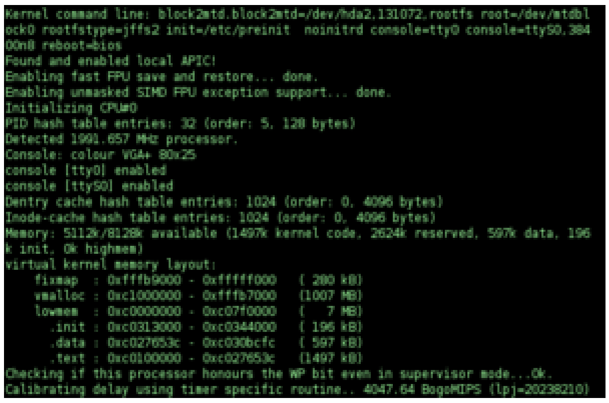
This topic was revisited in October 2008 by Amanda McPherson, Brian Proffitt, and Ron Hale-Evans. Using David A. Wheeler's methodology, they estimated redevelopment of the 2.6.25 kernel now costs $1.3bn (part of a total $10.8bn to redevelop Fedora 9).[277] Again, Garcia-Garcia and Alonso de Magdaleno 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.[278]

As of 7 March 2011, 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, 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 it would cost approximately $14,725,449,000 dollars (£11,191,341,000) to rewrite the existing code.[279]

Maintenance and long-term support

The latest kernel version and older kernel versions are maintained separately. Most latest kernel releases were supervised by Linus Torvalds.[280] Current versions are released by Greg Kroah-Hartman.[281]

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 will always list two stable kernels. The next stable Linux kernel is now released only 8 to 12 weeks later. Therefore, the Linux kernel maintainers have designated some stable kernel releases as *longterm*, these long-term support Linux kernels are updated with bug fixes for two or more years.[282] In November 2019 there were five longterm Linux kernels: 4.19.84, 4.14.154, 4.9.201, 4.4.201 and 3.16.76.[283] The full list of releases is at Linux kernel version history.



```
Kernel command line: block2mtd.block2mtd=/dev/hda2,131072,rootfs root=/dev/mtdbl
ock0 rootfstype=jffs2 init=/etc/preinit  noinitrd console=tty0 console=ttyS0,384
00n8 reboot=bios
Found and enabled local APIC!
Enabling fast FPU save and restore... done.
Enabling unmasked SIMD FPU exception support... done.
Initializing CPU#0
PID hash table entries: 32 (order: 5, 128 bytes)
Detected 1991.657 MHz processor.
Console: colour VGA+ 80x25
console [tty0] enabled
console [ttyS0] enabled
Dentry cache hash table entries: 1024 (order: 0, 4096 bytes)
Inode-cache hash table entries: 1024 (order: 0, 4096 bytes)
Memory: 5112k/8128k available (1497k kernel code, 2624k reserved, 597k data, 196
k init, 0k highmem)
virtual kernel memory layout:
    fixmap  : 0xfffb9000 - 0xfffff000   ( 280 kB)
    vmalloc : 0xc1000000 - 0xfffb7000   (1007 MB)
    lowmem  : 0xc0000000 - 0xc07f0000   (   7 MB)
      .init : 0xc0313000 - 0xc0344000   ( 196 kB)
      .data : 0xc027653c - 0xc030bcfc   ( 597 kB)
      .text : 0xc0100000 - 0xc027653c   (1497 kB)
Checking if this processor honours the WP bit even in supervisor mode...Ok.
Calibrating delay using timer specific routine.. 4047.64 BogoMIPS (lpj=20238210)
```

Boot messages of a Linux kernel 2.6.25.17

Relation with Linux distributions

Most Linux users run a kernel supplied by their Linux distribution. Some distributions ship the "vanilla" or "stable" kernels. However, several Linux distribution vendors (such as Red Hat and Debian) maintain another set of Linux kernel branches which are integrated into their products. These are usually updated at a slower pace compared to 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 their branch from.

Legal aspects

GPLv2 licensing terms

Initially, Torvalds released Linux under a license which forbade any commercial use.[284] This was changed in version 0.12 by a switch to the GNU General Public License version 2 (GPLv2).[27] 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 the complete corresponding source code.[285] Torvalds has described licensing Linux under the GPLv2 as the "best thing I ever did".[284]

The Linux kernel is licensed explicitly only under version 2 of the GPL,[22] without offering the licensee the option to choose "any later version", which is a common GPL extension. The official git branch of Torvalds contains documentation that explains the kernel development process to people who want to work with the community and contribute code; it clearly states that "[Any] contributions which are not covered by a [GPLv2] compatible license will not be accepted into the kernel.".[102]

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.[286] Torvalds himself specifically indicated upon the release of version 2.4.0 that his own code is released only under version 2.[287] However, the terms of the GPL state that if no version is specified, then any version may be used,[288] and Alan Cox pointed out that very few other Linux contributors had specified a particular version of the GPL.[289]

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."[290] 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.[291] They referred to clauses regarding DRM/tivoization, patents, "additional restrictions" and warned a Balkanisation of the "Open Source Universe" by the GPLv3.[291][292] Linus Torvalds, who decided not to adopt the GPLv3 for the Linux kernel, reiterated his criticism even years later.[293]

Loadable kernel modules

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[127][128] are non-derived works, thus Linux gives system administrators the mechanisms to load out-of-tree binary objects into the kernel address space.[4]

There are some out-of-tree loadable modules that make legitimate use of the *dma_buf* kernel feature.[294] 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.[129] Alan Cox replied on LKML, rejecting a request from one of their engineers to remove this technical enforcement from the API.[295] Torvalds clearly stated on the LKML that "[I] claim that binary-only kernel modules ARE derivative "by default"".[296]

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".[297] Proprietary graphics drivers, in particular, are heavily discussed.

Firmware binary blobs

The official kernel, that is the Linus git branch at the kernel.org repository, does not contain any kind of proprietary code;[22][4] however Linux can search the filesystems to locate proprietary firmware, drivers, and other executable modules (collectively known as "binary blobs"), then it can load and link them into the kernel space.[298] Whenever proprietary modules are loaded into Linux, the kernel marks itself as being "tainted",[299] and therefore bug reports from tainted kernels will often be ignored by developers.

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 firmware that is non-GPL compatible).[300]

Trademark

Linux is a registered trademark of Linus Torvalds in the United States, the European Union, and some other countries.[301][302] 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.[303][304][305]

See also

- Operating system
- Kernel
- Monolithic Kernel
- Microkernel
- Linux kernel version history
- Comparison of operating systems
- Comparison of operating system kernels
- Linux
- Minix 3
- macOS
- Microsoft Windows

References

1. "Linux Logos and Mascots" (https://web.archive.org/web/20100815085106/http://www.linux.org/info/logos.html). Linux Online. 2008. Archived from the original (http://www.linux.org/info/logos.html) on 15 August 2010. Retrieved 11 August 2009.
2. "Linux 5.13.12" (https://lore.kernel.org/lkml/162927267320059@kroah.com/). 18 August 2021. Retrieved 24 August 2021.
3. "Linux 5.14-rc7" (https://lore.kernel.org/lkml/CAHk-=wgZ_W7ZF84Mtq6KRjF4FEoYh14dnw+Oc0avZz_6WrCkfw@mail.gmail.com/). 22 August 2021. Retrieved 24 August 2021.
4. "Linux kernel licensing rules — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/process/license-rules.html#kernel-licensing). *www.kernel.org*. Archived (https://web.archive.org/web/20200307065451/https://www.kernel.org/doc/html/latest/process/license-rules.html#kernel-licensing) from the original on 7 March 2020. Retrieved 6 January 2020.
5. Tanenbaum, Andrew; Bos, Herbert (2015). *Modern Operating Systems*. United States of America: Pearson. p. 722. ISBN 9781292061429. OCLC 892574803 (https://www.worldcat.org/oclc/892574803).
6. Love, Robert (2010). *Linux kernel development*. Addison-Wesley. p. 4. ISBN 978-0-672-32946-3. OCLC 268788260 (https://www.worldcat.org/oclc/268788260).
7. Love, Robert (2010). *Linux kernel development*. Addison-Wesley. p. 338. ISBN 978-0-672-32946-3. OCLC 268788260 (https://www.worldcat.org/oclc/268788260).
8. "README" (https://archive.today/20120724163945/http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=README;h=90a07658ede14840346eee6610648bcf4ec79997;hb=f3b8436ad9a8ad36b3c9fa1fe030c7f38e5d3d0b). git.kernel.org. Archived from the original (https://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=README;h=90a07658ede14840346eee6610648bcf4ec79997;hb=f3b8436ad9a8ad36b3c9fa1fe030c7f38e5d3d0b) on 24 July 2012. Retrieved 24 March 2021.
9. Richardson, Marjorie (1 November 1999). "Interview: Linus Torvalds" (http://www.linuxjournal.com/article/3655). Linux Journal. Archived (https://web.archive.org/web/20110514084627/http://www.linuxjournal.com/article/3655) from the original on 14 May 2011. Retrieved 20 August 2009.

10. Williams, Sam (March 2002). "Chapter 9: The GNU General Public License" (https://archive.org/details/freeasinfreedomr00will). *Free as in Freedom: Richard Stallman's Crusade for Free Software*. O'Reilly. ISBN 0-596-00287-4. Retrieved 12 November 2010.
11. *Unix System Laboratories v. Berkeley Software*, 832 F. Supp. 790 (https://www.leagle.com/decision/19931622832fsupp79011506) (D.N.J. 1993).
12. "TOP500 Supercomputer Sites: Operating system Family / Linux" (https://www.top500.org/statistics/details/osfam/1). Top500.org. Archived (https://web.archive.org/web/20121119205719/https://www.top500.org/statistics/details/osfam/1) from the original on 19 November 2012. Retrieved 5 October 2019.
13. "Kernel Build System — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/kbuild/index.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200722122129/https://www.kernel.org/doc/html/latest/kbuild/index.html) from the original on 22 July 2020. Retrieved 17 July 2020.
14. "Kconfig make config — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/kbuild/kconfig.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200717132644/https://www.kernel.org/doc/html/latest/kbuild/kconfig.html) from the original on 17 July 2020. Retrieved 13 September 2020.
15. "KernelBuild - Linux Kernel Newbies" (https://kernelnewbies.org/KernelBuild). *kernelnewbies.org*. Archived (https://web.archive.org/web/20201019124650/https://kernelnewbies.org/KernelBuild) from the original on 19 October 2020. Retrieved 13 September 2020.
16. "The Sysctl Interface" (https://www.linux.it/~rubini/docs/sysctl/sysctl.html). *www.linux.it*. Archived (https://web.archive.org/web/20200217004812/http://www.linux.it/~rubini/docs/sysctl/sysctl.html) from the original on 17 February 2020. Retrieved 13 September 2020.
17. "sysctl(8) - Linux manual page" (https://man7.org/linux/man-pages/man8/sysctl.8.html). *man7.org*. Archived (https://web.archive.org/web/20200930200903/https://man7.org/linux/man-pages/man8/sysctl.8.html) from the original on 30 September 2020. Retrieved 13 September 2020.
18. "procfs(5) - Linux manual page" (https://man7.org/linux/man-pages/man5/procfs.5.html). *man7.org*. Archived (https://web.archive.org/web/20200924010905/https://man7.org/linux/man-pages/man5/procfs.5.html) from the original on 24 September 2020. Retrieved 13 September 2020.
19. Love, Robert (2010). *Linux Kernel Development*. Addison Wesley. p. 18. ISBN 978-0-672-32946-3. OCLC 268788260 (https://www.worldcat.org/oclc/268788260).
20. "C Extensions (Using the GNU Compiler Collection (GCC))" (https://gcc.gnu.org/onlinedocs/gcc-10.2.0/gcc/C-Extensions.html#C-Extensions). *gcc.gnu.org*. Archived (https://web.archive.org/web/20201120064908/https://gcc.gnu.org/onlinedocs/gcc-10.2.0/gcc/C-Extensions.html#C-Extensions) from the original on 20 November 2020. Retrieved 13 November 2020.
21. Love, Robert (2010). *Linux Kernel Development*. USA: Addison Wesley. pp. 379–380. ISBN 9780672329463.
22. "Linux source code: COPYING (v5.4.8) - Bootlin" (https://elixir.bootlin.com/linux/latest/source/COPYING). *elixir.bootlin.com*. Archived (https://web.archive.org/web/20200601173846/https://elixir.bootlin.com/linux/latest/source/COPYING) from the original on 1 June 2020. Retrieved 6 January 2020.
23. Torvalds, Linus Benedict (26 August 1991). "What would you like to see most in minix?" (https://groups.google.com/group/comp.os.minix/msg/b813d52cbc5a044b). Newsgroup: comp.os.minix (news:comp.os.minix). Usenet: 1991Aug25.205708.9541@klaava.Helsinki.FI (news:1991Aug25.205708.9541@klaava.Helsinki.FI). Archived (https://web.archive.org/web/20130509134305/http://groups.google.com/group/comp.os.minix/msg/b813d52cbc5a044b) from the original on 9 May 2013. Retrieved 14 September 2016.

24. Welsh, Matt; Dalheimer, Matthias Kalle; Kaufman, Lar (1999). "1". *Running Linux* (3rd ed.). Sebastopol, CA: O'Reilly Media, Inc. ISBN 1-56592-976-4. OCLC 50638246 (https://www.worldcat.org/oclc/50638246).
25. "Free minix-like kernel sources for 386-AT - Google Groups" (https://groups.google.com/g/comp.os.minix/c/4995SivOl9o/m/GwqLJlPSlCEJ). *groups.google.com*. 5 October 1991. Archived (https://web.archive.org/web/20210301162937/https://groups.google.com/g/comp.os.minix/c/4995SivOl9o/m/GwqLJlPSlCEJ) from the original on 1 March 2021. Retrieved 19 March 2020.
26. 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.
27. Torvalds, Linus. "Release Notes for Linux v0.12" (https://www.kernel.org/pub/linux/kernel/Historic/old-versions/RELNOTES-0.12). The Linux Kernel Archives. Archived (https://web.archive.org/web/20070819045030/http://www.kernel.org/pub/linux/kernel/Historic/old-versions/RELNOTES-0.12) from the original on 19 August 2007. Retrieved 21 February 2007.
28. Fred Hantelmann (2016). *LINUX Start-up Guide: A self-contained introduction*. Springer Science & Business Media. p. 1. ISBN 9783642607493.
29. Fred Hantelmann (2016). *LINUX Start-up Guide: A self-contained introduction*. Springer Science & Business Media. p. 16. ISBN 9783642607493.
30. Summers, David W. (19 January 1992). "Troubles with Partitions" (https://groups.google.com/group/alt.os.linux/msg/c638df159fa15159). Newsgroup: alt.os.linux (news:alt.os.linux). Usenet: 1992Jan19.085628.18752@cseg01.uark.edu (news:1992Jan19.085628.18752@cseg01.uark.edu). Archived (https://web.archive.org/web/20130602210415/http://groups.google.com/group/alt.os.linux/msg/c638df159fa15159) from the original on 2 June 2013. Retrieved 7 January 2007.
31. Clegg, Alan B. (31 March 1992). "It's here!" (https://groups.google.com/group/comp.os.linux/msg/81fe3618c4803d1e). Newsgroup: comp.os.linux (news:comp.os.linux). Usenet: 1992Mar31.131811.19832@rock.concert.net (news:1992Mar31.131811.19832@rock.concert.net). Archived (https://web.archive.org/web/20130602203914/http://groups.google.com/group/comp.os.linux/msg/81fe3618c4803d1e) from the original on 2 June 2013. Retrieved 7 January 2007.
32. "Appendix A: The Tanenbaum-Torvalds Debate" (https://archive.org/details/isbn_9781565925823). *Open Sources: Voices from the Open Source Revolution*. O'Reilly. 1999. ISBN 1-56592-582-3. Retrieved 22 November 2006.
33. Tanenbaum, Andy (29 January 1992). "LINUX is obsolete" (https://groups.google.com/group/comp.os.minix/msg/f447530d082cd95d). Newsgroup: comp.os.minix (news:comp.os.minix). Usenet: 12595@star.cs.vu.nl (news:12595@star.cs.vu.nl). Archived (https://web.archive.org/web/20111017163006/http://groups.google.com/group/comp.os.minix/msg/f447530d082cd95d) from the original on 17 October 2011. Retrieved 10 May 2006.
34. Tanenbaum, Andy (12 May 2006). "Tanenbaum-Torvalds Debate: Part II" (http://www.cs.vu.nl/~ast/reliable-os/). VU University Amsterdam. Archived (https://web.archive.org/web/20150805132304/http://www.cs.vu.nl/~ast/reliable-os/) from the original on 5 August 2015. Retrieved 6 January 2007.
35. November 2012, David Hayward22 (22 November 2012). "The history of Linux: how time has shaped the penguin" (https://www.techradar.com/news/software/operating-systems/the-history-of-linux-how-time-has-shaped-the-penguin-1113914). *TechRadar*. Archived (https://web.archive.org/web/20200319065513/https://www.techradar.com/news/software/operating-systems/the-history-of-linux-how-time-has-shaped-the-penguin-1113914) from the original on 19 March 2020. Retrieved 19 March 2020.

36. November 2012, David Hayward22 (22 November 2012). "The history of Linux: how time has shaped the penguin" (https://www.techradar.com/news/software/operating-systems/the-history-of-linux-how-time-has-shaped-the-penguin-1113914/2). *TechRadar*. Archived (https://web.archive.org/web/20200319065522/https://www.techradar.com/news/software/operating-systems/the-history-of-linux-how-time-has-shaped-the-penguin-1113914/2) from the original on 19 March 2020. Retrieved 26 March 2020.
37. Love, Robert (Robert M.) (2010). *Linux kernel development* (3rd ed.). Upper Saddle River, NJ: Addison-Wesley. p. 9. ISBN 978-0-672-32946-3. OCLC 268788260 (https://www.worldcat.org/oclc/268788260).
38. "How the development process works — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/process/2.Process.html#the-big-picture). *www.kernel.org*. Archived (https://web.archive.org/web/20171209130758/https://www.kernel.org/doc/html/latest/process/2.Process.html#the-big-picture) from the original on 9 December 2017. Retrieved 26 March 2020.
39. 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.
40. Torvalds, Linus (9 June 1996). "Linux 2.0 really _is_ released." (http://lkml.iu.edu/hypermail/linux/kernel/9606.1/0056.html) *LKML* (Mailing list). Archived (https://web.archive.org/web/20150402091044/http://lkml.iu.edu/hypermail/linux/kernel/9606.1/0056.html) from the original on 2 April 2015. Retrieved 8 March 2015.
41. Torvalds, Linus (20 January 1999). "2.2.0-final" (http://lkml.iu.edu/hypermail/linux/kernel/9901.2/1084.html). *LKML* (Mailing list). Archived (https://web.archive.org/web/20150402144000/http://lkml.iu.edu/hypermail/linux/kernel/9901.2/1084.html) from the original on 2 April 2015. Retrieved 8 March 2015.
42. "The Wonderful World of Linux 2.2" (http://kniggit.net/wonderful-world-linux/wonderful-world-linux-2-2/). 26 January 1999. Archived (https://web.archive.org/web/20141106030845/http://kniggit.net/wonderful-world-linux/wonderful-world-linux-2-2/) from the original on 6 November 2014. Retrieved 27 October 2008.
43. "Linux/390 Observations and Notes" (http://linuxvm.org/penguinvm/notes.html). *linuxvm.org*. Archived (https://web.archive.org/web/20190226085302/http://linuxvm.org/penguinvm/notes.html) from the original on 26 February 2019. Retrieved 29 March 2020.
44. Torvalds, Linus (4 January 2001). "And oh, btw." (http://lkml.iu.edu/hypermail/linux/kernel/0101.0/0776.html) *LKML* (Mailing list). Archived (https://web.archive.org/web/20160126231619/http://lkml.iu.edu/hypermail/linux/kernel/0101.0/0776.html) from the original on 26 January 2016. Retrieved 8 March 2015.
45. "The Wonderful World of Linux 2.4" (https://web.archive.org/web/20050317071343/http://www.kniggit.net/wwol24.html). Archived from the original (http://kniggit.net/wwol24.html) on 17 March 2005. Retrieved 27 October 2008.
46. Torvalds, Linus (17 December 2003). "Linux 2.6.0" (http://lkml.iu.edu/hypermail/linux/kernel/0312.2/0348.html). *LKML* (Mailing list). Archived (https://web.archive.org/web/20150402162542/http://lkml.iu.edu/hypermail/linux/kernel/0312.2/0348.html) from the original on 2 April 2015. Retrieved 28 February 2015.
47. "proc(5) - Linux manual page" (http://man7.org/linux/man-pages/man5/proc.5.html) (see /proc/sys/kernel/pid_max). Archived (https://web.archive.org/web/20140207232837/http://man7.org/linux/man-pages/man5/proc.5.html) from the original on 7 February 2014. Retrieved 19 February 2014.
48. "btrfs Wiki" (https://btrfs.wiki.kernel.org/index.php/Main_Page). *btrfs.wiki.kernel.org*. Archived (https://web.archive.org/web/20120425151829/https://btrfs.wiki.kernel.org/) from the original on 25 April 2012. Retrieved 17 July 2020.
49. Fred Hantelmann (2016). *LINUX Start-up Guide: A self-contained introduction*. Springer Science & Business Media. pp. 1–2. ISBN 9783642607493.

50. Kroah-Hartman, Greg (3 August 2006). "Adrian Bunk is now taking over the 2.6.16-stable branch" (http://lkml.iu.edu/hypermail/linux/kernel/0608.0/1111.html). *LKML* (Mailing list). Archived (https://web.archive.org/web/20160126231617/http://lkml.iu.edu/hypermail/linux/kernel/0608.0/1111.html) from the original on 26 January 2016. Retrieved 21 February 2015.
51. Rothwell, Stephen (12 February 2008). "Announce: Linux-next (Or Andrew's dream :-))" (https://lkml.org/lkml/2008/2/11/512). *LKML* (Mailing list). Archived (https://web.archive.org/web/20101124235700/http://lkml.org/lkml/2008/2/11/512) from the original on 24 November 2010. Retrieved 30 October 2010.
52. Corbet, Jonathan (21 October 2010). "linux-next and patch management process" (https://lwn.net/Articles/269120/). *LWN.net*. Eklektix, Inc. Archived (https://web.archive.org/web/20100621034215/http://lwn.net/Articles/269120/) from the original on 21 June 2010. Retrieved 30 October 2010.
53. "The Linux Kernel Archives" (http://www.kernel.org). Kernel.org. Archived (https://web.archive.org/web/19980130085039/http://www.kernel.org/) from the original on 30 January 1998. Retrieved 22 January 2014.
54. Linux Kernel Mailing List (17 June 2005). "Linux 2.6.12" (https://marc.info/?l=git-commits-head&m=111904216911731). *git-commits-head* (Mailing list). Archived (https://web.archive.org/web/20160126231629/http://marc.info/?l=git-commits-head&m=111904216911731) from the original on 26 January 2016. Retrieved 23 January 2008.
55. "Index of /pub/linux/kernel/v2.6" (https://www.kernel.org/pub/linux/kernel/v2.6/). Kernel.org. Archived (https://web.archive.org/web/20140210131743/https://www.kernel.org/pub/linux/kernel/v2.6/) from the original on 10 February 2014. Retrieved 2 March 2014.
56. "Add a personality to report 2.6.x version numbers [LWN.net]" (https://lwn.net/Articles/451168/). *lwn.net*. Archived (https://web.archive.org/web/20200716092939/https://lwn.net/Articles/451168/) from the original on 16 July 2020. Retrieved 15 July 2020.
57. Torvalds, Linus (21 July 2011). "Linux 3.0 release" (http://lkml.indiana.edu/hypermail/linux/kernel/1107.2/01843.html). Linux kernel mailing list. Archived (https://web.archive.org/web/20191018044641/http://lkml.iu.edu/hypermail/linux/kernel/1107.2/01843.html) from the original on 18 October 2019. Retrieved 16 May 2013.
58. Torvalds, Linus (30 May 2011). "Linux 3.0-rc1" (https://web.archive.org/web/20110531232747/http://permalink.gmane.org/gmane.linux.kernel/1147415). *LKML* (Mailing list). Archived from the original (http://permalink.gmane.org/gmane.linux.kernel/1147415) on 31 May 2011. Retrieved 1 July 2013.
59. Vaughan-Nichols, Steven J. (13 December 2012). "Good-Bye 386: Linux to drop support for i386 chips with next major release" (https://www.zdnet.com/article/good-bye-386-linux-to-drop-support-for-i386-chips-with-next-major-release/). *ZDNet*. CBS Interactive. Archived (https://web.archive.org/web/20150217232706/http://www.zdnet.com/article/good-bye-386-linux-to-drop-support-for-i386-chips-with-next-major-release/) from the original on 17 February 2015. Retrieved 6 February 2013.
60. Fingas, Jon (15 December 2012). "Linux to drop i386 support in the 3.8 kernel, make us upgrade our Doom rig" (https://www.engadget.com/2012/12/15/linux-to-drop-i386-support-in-the-3-8-kernel/). *Engadget*. AOL. Archived (https://web.archive.org/web/20150402141004/http://www.engadget.com/2012/12/15/linux-to-drop-i386-support-in-the-3-8-kernel/) from the original on 2 April 2015. Retrieved 22 March 2015.
61. Vaughan-Nichols, Steven J. (11 December 2012). "Linux 3.7 arrives, ARM developers rejoice" (https://www.zdnet.com/linux-3-7-arrives-arm-developers-rejoice-7000008638/). *ZDNet*. CBS Interactive. Archived (https://web.archive.org/web/20141105164320/http://www.zdnet.com/linux-3-7-arrives-arm-developers-rejoice-7000008638/) from the original on 5 November 2014. Retrieved 6 February 2013.

62. Torvalds, Linus (2 September 2013). "Linux 3.11" (http://lkml.iu.edu/hypermail/linux/kernel/1309.0/00650.html). *LKML* (Mailing list). Archived (https://web.archive.org/web/20140226021932/http://lkml.iu.edu//hypermail/linux/kernel/1309.0/00650.html) from the original on 26 February 2014. Retrieved 3 September 2013.
63. "Linux 3.11" (http://kernelnewbies.org/Linux_3.11). kernelnewbies.org. 2 September 2013. Retrieved 21 January 2014.
64. Torvalds, Linus (12 April 2015). "Linux 4.0 released" (https://lkml.org/lkml/2015/4/12/178). *LKML* (Mailing list). Archived (https://web.archive.org/web/20150413015619/https://lkml.org/lkml/2015/4/12/178) from the original on 13 April 2015. Retrieved 12 April 2015.
65. "The Linux Foundation Releases Linux Development Report" (https://web.archive.org/web/20160719042639/https://www.linuxfoundation.org/news-media/announcements/2015/02/linux-foundation-releases-linux-development-report). Linux Foundation. 18 February 2015. Archived from the original (http://www.linuxfoundation.org/news-media/announcements/2015/02/linux-foundation-releases-linux-development-report) on 19 July 2016. Retrieved 20 February 2015.
66. Michael Larabel (23 June 2014). "Linux Kernel At 19.5 Million Lines Of Code, Continues Rising" (https://www.phoronix.com/scan.php?page=news_item&px=Linux-19.5M-Stats). Phoronix. Archived (https://web.archive.org/web/20201123170810/https://www.phoronix.com/scan.php?page=news_item&px=Linux-19.5M-Stats) from the original on 23 November 2020. Retrieved 23 June 2015.
67. Corbet, Jonathan (3 August 2020). "Some statistics from the 5.8 kernel cycle" (https://lwn.net/Articles/827735/). *LWN - Linux Weekly News*. Archived (https://web.archive.org/web/20200904084101/https://lwn.net/Articles/827735/) from the original on 4 September 2020. Retrieved 11 August 2020.
68. "Stack Overflow Developer Survey 2019 - most popular technologies" (https://insights.stackoverflow.com/survey/2019/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2019). *Stack Overflow*. Archived (https://web.archive.org/web/20201008033536/https://insights.stackoverflow.com/survey/2019/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2019) from the original on 8 October 2020. Retrieved 17 March 2020.
69. "Stack Overflow Developer Survey 2019 - development environments and tools" (https://insights.stackoverflow.com/survey/2019#development-environments-and-tools). *Stack Overflow*. Archived (https://web.archive.org/web/20200307082721/https://insights.stackoverflow.com/survey/2019#development-environments-and-tools) from the original on 7 March 2020. Retrieved 17 March 2020.
70. "Usage Statistics and Market Share of Operating Systems for Websites, March 2020" (https://w3techs.com/technologies/overview/operating_system). *w3techs.com*. Retrieved 17 March 2020.
71. "Usage Statistics and Market Share of Unix for Websites, March 2020" (https://w3techs.com/technologies/details/os-unix). *w3techs.com*. Retrieved 17 March 2020.
72. "TOP500 Supercomputer Sites: Operating system Family / Linux" (https://www.top500.org/statistics/details/osfam/1). Top500.org. Archived (https://web.archive.org/web/20121119205719/https://www.top500.org/statistics/details/osfam/1) from the original on 19 November 2012. Retrieved 5 October 2019.
73. "Gartner Says Sales of Tablets Will Represent Less Than 10 Percent of All Devices in 2014" (http://www.gartner.com/newsroom/id/2875017) (Press release). Egham, UK: Gartner. 15 October 2014. Archived (https://web.archive.org/web/20141017151529/http://www.gartner.com/newsroom/id/2875017) from the original on 17 October 2014. Retrieved 19 October 2014.

74. Lunden, Ingrid (15 October 2014). "Tablet Sales Growth Plummets In 2014 As Android Smartphones Continue To Soar: Gartner" (https://techcrunch.com/2014/10/15/tablet-sales-growth-plummets-in-2014-as-android-smartphones-continue-to-soar-gartner). *TechCrunch*. AOL. Archived (https://web.archive.org/web/20141023114800/http://techcrunch.com/2014/10/15/tablet-sales-growth-plummets-in-2014-as-android-smartphones-continue-to-soar-gartner/) from the original on 23 October 2014. Retrieved 23 October 2014.
75. "Global PC Shipments Exceed Forecast with Mild Improvement in Consumer Demand, While Apple Moves to #5 Spot, According to IDC" (https://web.archive.org/web/20141011215307/http://www.idc.com/getdoc.jsp?containerId=prUS25187214) (Press release). Framingham, MA: IDC. 8 October 2014. Archived from the original (http://www.idc.com/getdoc.jsp?containerId=prUS25187214) on 11 October 2014. Retrieved 19 October 2014.
76. "sched(7) - Linux manual page" (https://man7.org/linux/man-pages/man7/sched.7.html). *man7.org*. Archived (https://web.archive.org/web/20200717155549/https://man7.org/linux/man-pages/man7/sched.7.html) from the original on 17 July 2020. Retrieved 27 July 2020.
77. "FAQ: Preemption" (http://kernelnewbies.org/FAQ/Preemption). *kernelnewbies.org*. 22 August 2009. Archived (https://web.archive.org/web/20200807081640/https://kernelnewbies.org/FAQ/Preemption) from the original on 7 August 2020. Retrieved 7 May 2015.
78. Jonathan Corbet (24 February 2003). "Driver porting: the preemptible kernel" (https://lwn.net/Articles/22912/). LWN.net. Archived (https://web.archive.org/web/20200810170137/https://lwn.net/Articles/22912/) from the original on 10 August 2020. Retrieved 7 May 2015.
79. Molnár, Ingo (13 April 2007). "[patch] Modular Scheduler Core and Completely Fair Scheduler [CFS]" (https://lwn.net/Articles/230501/). *LKML* (Mailing list). Archived (https://web.archive.org/web/20201103034312/https://lwn.net/Articles/230501/) from the original on 3 November 2020. Retrieved 30 March 2020.
80. "Completely Fair Scheduler | Linux Journal" (https://www.linuxjournal.com/node/10267). *www.linuxjournal.com*. Archived (https://web.archive.org/web/20200803104512/https://www.linuxjournal.com/node/10267) from the original on 3 August 2020. Retrieved 30 March 2020.
81. "ioctl(2) - Linux manual page" (https://man7.org/linux/man-pages/man2/ioctl.2.html). *man7.org*. Archived (https://web.archive.org/web/20200720073257/https://man7.org/linux/man-pages/man2/ioctl.2.html) from the original on 20 July 2020. Retrieved 11 August 2020.
82. "aio(7) - Linux manual page" (https://man7.org/linux/man-pages/man7/aio.7.html). *man7.org*. Archived (https://web.archive.org/web/20200412005208/http://man7.org/linux/man-pages/man7/aio.7.html) from the original on 12 April 2020. Retrieved 11 August 2020.
83. "io_setup(2) - Linux manual page" (https://man7.org/linux/man-pages/man2/io_setup.2.html). *man7.org*. Archived (https://web.archive.org/web/20200820190947/https://man7.org/linux/man-pages/man2/io_setup.2.html) from the original on 20 August 2020. Retrieved 11 August 2020.
84. "KVM" (https://www.linux-kvm.org/page/Main_Page). *www.linux-kvm.org*. Archived (https://web.archive.org/web/20200328192644/https://www.linux-kvm.org/page/Main_Page) from the original on 28 March 2020. Retrieved 29 March 2020.
85. "TechComparison - Linux Virtualization Wiki" (https://virt.kernelnewbies.org/TechComparison). *virt.kernelnewbies.org*. Archived (https://web.archive.org/web/20200803081859/https://virt.kernelnewbies.org/TechComparison) from the original on 3 August 2020. Retrieved 29 March 2020.
86. "Virtualization_support_through_KVM in Linux_2_6_20 - Linux Kernel Newbies" (https://kernelnewbies.org/Linux_2_6_20#Virtualization_support_through_KVM). *kernelnewbies.org*. Archived (https://web.archive.org/web/20191129072053/https://kernelnewbies.org/Linux_2_6_20#Virtualization_support_through_KVM) from the original on 29 November 2019. Retrieved 29 March 2020.

87. Coekaerts, Wim. "Linux mainline contains all the Xen code bits for Dom0 and DomU support" (https://blogs.oracle.com/wim/linux-mainline-contains-all-the-xen-code-bits-for-dom0-and-domu-support). *blogs.oracle.com*. Archived (https://web.archive.org/web/20200803103832/https://blogs.oracle.com/wim/linux-mainline-contains-all-the-xen-code-bits-for-dom0-and-domu-support) from the original on 3 August 2020. Retrieved 29 March 2020.
88. "Xen celebrates full Dom0 and DomU support in Linux 3.0 – blog.xen.org" (https://web.archive.org/web/20110607003740/http://blog.xen.org/index.php/2011/06/02/xen-celebrates-full-dom0-and-domu-support-in-linux-3-0/). 7 June 2011. Archived from the original (http://blog.xen.org/index.php/2011/06/02/xen-celebrates-full-dom0-and-domu-support-in-linux-3-0/) on 7 June 2011. Retrieved 29 March 2020.
89. Wilk, Konrad Rzeszutek (31 January 2014). "Linux 3.14 and PVH" (https://xenproject.org/2014/01/31/linux-3-14-and-pvh/). *Xen Project*. Archived (https://web.archive.org/web/20200329115320/https://xenproject.org/2014/01/31/linux-3-14-and-pvh/) from the original on 29 March 2020. Retrieved 29 March 2020.
90. "Introduction to Xen Virtualization | Virtualization Guide | openSUSE Leap 15.2" (https://doc.opensuse.org/documentation/leap/virtualization/html/book.virt/cha-xen-basics.html). *doc.opensuse.org*. Archived (https://web.archive.org/web/20200928214033/https://doc.opensuse.org/documentation/leap/virtualization/html/book.virt/cha-xen-basics.html) from the original on 28 September 2020. Retrieved 29 September 2020.
91. "SELinux Project" (https://github.com/SELinuxProject). *GitHub*. Archived (https://web.archive.org/web/20191212214729/https://github.com/SELinuxProject) from the original on 12 December 2019. Retrieved 10 January 2020.
92. "AppArmor — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/admin-guide/LSM/apparmor.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200508080035/https://www.kernel.org/doc/html/latest/admin-guide/LSM/apparmor.html) from the original on 8 May 2020. Retrieved 10 January 2020.
93. Jake Edge (25 November 2008). "Character devices in user space" (https://lwn.net/Articles/308445/). LWN.net. Archived (https://web.archive.org/web/20210126131908/https://lwn.net/Articles/308445/) from the original on 26 January 2021. Retrieved 7 May 2015.
94. Jonathan Corbet (2 May 2007). "UIO: user-space drivers" (https://lwn.net/Articles/232575/). LWN.net. Archived (https://web.archive.org/web/20201111193009/https://lwn.net/Articles/232575/) from the original on 11 November 2020. Retrieved 7 May 2015.
95. "stable-api-nonsense - Linux kernel source tree" (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/process/stable-api-nonsense.rst). *git.kernel.org*. Archived (https://web.archive.org/web/20210305010734/https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/process/stable-api-nonsense.rst) from the original on 5 March 2021. Retrieved 18 April 2020.
96. Gorman, Mel (15 February 2004). *Understanding the Linux Virtual Memory Manager* (https://pdos.csail.mit.edu/~sbw/links/gorman_book.pdf) (PDF). Prentice Hall. p. 26. ISBN 0-13-145348-3. Archived (https://web.archive.org/web/20190503113248/https://pdos.csail.mit.edu/~sbw/links/gorman_book.pdf) (PDF) from the original on 3 May 2019. Retrieved 27 January 2020.
97. Greg Ungerer. "uClinux mainline Announcement" (https://web.archive.org/web/20071031135123/http://www.ucdot.org/article.pl?sid=02%2F11%2F05%2F0324207). Archived from the original (http://www.ucdot.org/article.pl?sid=02/11/05/0324207) on 31 October 2007. Retrieved 15 January 2008.
98. Nguyen, Binh (30 July 2004). "Linux Filesystem Hierarchy: Chapter 1. Linux Filesystem Hierarchy" (http://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/index.html). The Linux Documentation Project. Archived (https://web.archive.org/web/20201202064950/https://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/index.html) from the original on 2 December 2020. Retrieved 28 November 2012.

99. "Linux kernel release 5.x — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/admin-guide/README.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200307065108/https://www.kernel.org/doc/html/latest/admin-guide/README.html) from the original on 7 March 2020. Retrieved 4 January 2020.
100. "README\ABI\Documentation - kernel/git/torvalds/linux.git - Linux kernel source tree" (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/ABI/README). *git.kernel.org*. Archived (https://web.archive.org/web/20201001172809/https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/ABI/README) from the original on 1 October 2020. Retrieved 18 April 2020.
101. "syscalls\stable\ABI\Documentation - kernel/git/torvalds/linux.git - Linux kernel source tree" (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/ABI/stable/syscalls). *git.kernel.org*. Archived (https://web.archive.org/web/20201002061451/https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/ABI/stable/syscalls) from the original on 2 October 2020. Retrieved 18 April 2020.
102. "1.Intro.rst - Documentation/process/1.Intro.rst - Linux source code (v5.8) - Bootlin" (https://elixir.bootlin.com/linux/latest/source/Documentation/process/1.Intro.rst). *elixir.bootlin.com*. Retrieved 8 August 2020.
103. "syscalls" (http://man7.org/linux/man-pages/man2/syscalls.2.html). *man7*. Archived (https://web.archive.org/web/20200115033131/http://man7.org/linux/man-pages/man2/syscalls.2.html) from the original on 15 January 2020. Retrieved 28 January 2020.
104. "intro(2) - Linux manual page" (https://man7.org/linux/man-pages/man2/intro.2.html). *man7.org*. Archived (https://web.archive.org/web/20200717161934/https://man7.org/linux/man-pages/man2/intro.2.html) from the original on 17 July 2020. Retrieved 16 July 2020.
105. "clone" (http://man7.org/linux/man-pages/man2/clone.2.html). *man7.org*. Archived (https://web.archive.org/web/20200118015900/http://man7.org/linux/man-pages/man2/clone.2.html) from the original on 18 January 2020. Retrieved 28 January 2020.
106. "feature_test_macros" (http://man7.org/linux/man-pages/man7/feature_test_macros.7.html). *man7.org*. Archived (https://web.archive.org/web/20200119174511/http://man7.org/linux/man-pages/man7/feature_test_macros.7.html) from the original on 19 January 2020. Retrieved 28 January 2020.
107. "vdso(7) - Linux manual page" (http://man7.org/linux/man-pages/man7/vdso.7.html). *man7.org*. Archived (https://web.archive.org/web/20200202123949/http://man7.org/linux/man-pages/man7/vdso.7.html) from the original on 2 February 2020. Retrieved 2 February 2020.
108. "futex(2) - Linux manual page" (http://man7.org/linux/man-pages/man2/futex.2.html). *man7.org*. Archived (https://web.archive.org/web/20200131144454/http://man7.org/linux/man-pages/man2/futex.2.html) from the original on 31 January 2020. Retrieved 2 February 2020.
109. "syscall(2) - Linux manual page" (http://man7.org/linux/man-pages/man2/syscall.2.html). *man7.org*. Archived (https://web.archive.org/web/20200121174524/http://man7.org/linux/man-pages/man2/syscall.2.html) from the original on 21 January 2020. Retrieved 2 February 2020.
110. "sysfs(5) - Linux manual page" (http://man7.org/linux/man-pages/man5/sysfs.5.html). *man7.org*. Archived (https://web.archive.org/web/20200118044323/http://man7.org/linux/man-pages/man5/sysfs.5.html) from the original on 18 January 2020. Retrieved 6 January 2020.
111. "Rules on how to access information in sysfs — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/admin-guide/sysfs-rules.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200307065123/https://www.kernel.org/doc/html/latest/admin-guide/sysfs-rules.html) from the original on 7 March 2020. Retrieved 6 January 2020.
112. "Linux Foundation Referenced Specifications" (https://refspecs.linuxbase.org/). *refspecs.linuxbase.org*. Retrieved 3 February 2020.

113. "LSB Specifications" (https://refspecs.linuxbase.org/lsb.shtml). *refspecs.linuxbase.org*. Retrieved 3 February 2020.
114. "Linux Standard Base Desktop Specification, Generic Part" (https://refspecs.linuxbase.org/LSB_5.0.0/LSB-Desktop-generic/LSB-Desktop-generic/book1.html). *refspecs.linuxbase.org*. Retrieved 3 February 2020.
115. "Normative References" (https://refspecs.linuxfoundation.org/LSB_5.0.0/LSB-Core-generic/LSB-Core-generic/normativerefs.html). *refspecs.linuxfoundation.org*. Archived (https://web.archive.org/web/20200812044159/https://refspecs.linuxfoundation.org/LSB_5.0.0/LSB-Core-generic/LSB-Core-generic/normativerefs.html) from the original on 12 August 2020. Retrieved 3 February 2020.
116. "Linux Standard Base Core Specification, Generic Part" (https://refspecs.linuxfoundation.org/LSB_5.0.0/LSB-Core-generic/LSB-Core-generic/book1.html). *refspecs.linuxfoundation.org*. Archived (https://web.archive.org/web/20191129194815/https://refspecs.linuxfoundation.org/LSB_5.0.0/LSB-Core-generic/LSB-Core-generic/book1.html) from the original on 29 November 2019. Retrieved 3 February 2020.
117. "System V Application Binary Interface - Edition 4.1" (https://www.sco.com/developers/devspecs/gabi41.pdf) (PDF). *www.sco.com*. Archived (https://web.archive.org/web/20191213124815/http://www.sco.com/developers/devspecs/gabi41.pdf) (PDF) from the original on 13 December 2019. Retrieved 3 February 2020.
118. "Xinuos Inc. | Developers | Gabi | 2003-12-17 | System V Application Binary Interface - DRAFT" (http://www.sco.com/developers/gabi/2003-12-17/contents.html). *www.sco.com*. Archived (https://web.archive.org/web/20200203124116/http://www.sco.com/developers/gabi/2003-12-17/contents.html) from the original on 3 February 2020. Retrieved 3 February 2020.
119. "Executable And Linking Format (ELF)" (https://refspecs.linuxbase.org/LSB_5.0.0/LSB-Core-generic/LSB-Core-generic/elf-generic.html). *refspecs.linuxbase.org*. Retrieved 3 February 2020.
120. "elf(5) - Linux manual page" (https://man7.org/linux/man-pages/man5/elf.5.html). *man7.org*. Archived (https://web.archive.org/web/20201130114725/https://man7.org/linux/man-pages/man5/elf.5.html) from the original on 30 November 2020. Retrieved 18 November 2020.
121. "Linux Standard Base Core Specification for X86-64" (https://refspecs.linuxbase.org/LSB_5.0.0/LSB-Core-AMD64/LSB-Core-AMD64/book1.html). *refspecs.linuxbase.org*. Retrieved 3 February 2020.
122. "System V Application Binary Interface - DRAFT" (https://refspecs.linuxbase.org/elf/gabi4+/contents.html). *refspecs.linuxbase.org*. Retrieved 3 February 2020.
123. Seyfarth, Ray (2012). *Introduction to 64 Bit Intel Assembly Language Programming for Linux*. p. 170. ISBN 9781478119203.
124. "Anatomy of a system call, part 1 [LWN.net]" (https://lwn.net/Articles/604287/). *lwn.net*. Archived (https://web.archive.org/web/20200818051836/https://lwn.net/Articles/604287/) from the original on 18 August 2020. Retrieved 16 July 2020.
125. "Anatomy of a system call, part 2 [LWN.net]" (https://lwn.net/Articles/604515/). *lwn.net*. Archived (https://web.archive.org/web/20200806081538/https://lwn.net/Articles/604515/) from the original on 6 August 2020. Retrieved 16 July 2020.
126. Deucher, Alex (7 October 2014). "AMD's New Unified Open Source Driver" (http://wiki.x.org/wiki/Events/XDC2014/XDC2014DeucherAMD/). X.Org Foundation. Archived (https://web.archive.org/web/20150121163629/http://wiki.x.org/wiki/Events/XDC2014/XDC2014DeucherAMD/) from the original on 21 January 2015. Retrieved 21 January 2015.
127. "Symbols - Unreliable Guide To Hacking The Linux Kernel — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/kernel-hacking/hacking.html?highlight=export_symbol#symbols). *www.kernel.org*. Archived (https://web.archive.org/web/20200803074501/https://www.kernel.org/doc/html/latest/kernel-hacking/hacking.html?highlight=export_symbol#symbols) from the original on 3 August 2020. Retrieved 8 February 2020.

128. "Exported symbols and the internal API [LWN.net]" (https://lwn.net/Articles/249246/). *lwn.net*. Archived (https://web.archive.org/web/20200331211446/https://lwn.net/Articles/249246/) from the original on 31 March 2020. Retrieved 15 March 2020.
129. "Unexporting kallsyms_lookup_name() [LWN.net]" (https://lwn.net/Articles/813350/). *lwn.net*. Archived (https://web.archive.org/web/20200401062303/https://lwn.net/Articles/813350/) from the original on 1 April 2020. Retrieved 15 March 2020.
130. "Trees I: Radix trees [LWN.net]" (https://lwn.net/Articles/175432/). *lwn.net*. Archived (https://web.archive.org/web/20201108131647/https://lwn.net/Articles/175432/) from the original on 8 November 2020. Retrieved 13 November 2020.
131. "Trees II: red-black trees [LWN.net]" (https://lwn.net/Articles/184495/). *lwn.net*. Archived (https://web.archive.org/web/20201113130357/https://lwn.net/Articles/184495/) from the original on 13 November 2020. Retrieved 13 November 2020.
132. "Unreliable Guide To Hacking The Linux Kernel" (https://www.kernel.org/doc/htmldocs/kernel-hacking/index.html). *www.kernel.org* (1st ed.). 2005. Archived (https://web.archive.org/web/20200216191225/https://www.kernel.org/doc/htmldocs/kernel-hacking/index.html) from the original on 16 February 2020. Retrieved 15 March 2020.
133. "Unreliable Guide To Hacking The Linux Kernel — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/kernel-hacking/hacking.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200307065323/https://www.kernel.org/doc/html/latest/kernel-hacking/hacking.html) from the original on 7 March 2020. Retrieved 15 March 2020.
134. "Unreliable Guide To Locking — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/kernel-hacking/locking.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200307065319/https://www.kernel.org/doc/html/latest/kernel-hacking/locking.html) from the original on 7 March 2020. Retrieved 15 March 2020.
135. "SCSI Interfaces Guide — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/driver-api/scsi.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200602154450/https://www.kernel.org/doc/html/latest/driver-api/scsi.html) from the original on 2 June 2020. Retrieved 11 June 2020.
136. "libATA Developer's Guide — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/driver-api/libata.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200530101401/https://www.kernel.org/doc/html/latest/driver-api/libata.html) from the original on 30 May 2020. Retrieved 11 June 2020.
137. "DRM Internals — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/gpu/drm-internals.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200601202717/https://www.kernel.org/doc/html/latest/gpu/drm-internals.html) from the original on 1 June 2020. Retrieved 11 June 2020.
138. "Kernel Mode Setting (KMS) — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/gpu/drm-kms.html#overview). *www.kernel.org*. Archived (https://web.archive.org/web/20200611233817/https://www.kernel.org/doc/html/latest/gpu/drm-kms.html#overview) from the original on 11 June 2020. Retrieved 11 June 2020.
139. "Introduce DMA buffer sharing mechanism [LWN.net]" (https://lwn.net/Articles/473668/). *lwn.net*. Archived (https://web.archive.org/web/20200611235759/https://lwn.net/Articles/473668/) from the original on 11 June 2020. Retrieved 11 June 2020.
140. "Sharing CPU and GPU buffers on Linux*" (https://01.org/blogs/2016/sharing-cpu-and-gpu-buffers-linux). *01.org*. 12 May 2016. Archived (https://web.archive.org/web/20200611231858/https://01.org/blogs/2016/sharing-cpu-and-gpu-buffers-linux) from the original on 11 June 2020. Retrieved 11 June 2020.
141. "Buffer Sharing and Synchronization — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/driver-api/dma-buf.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200601205610/https://www.kernel.org/doc/html/latest/driver-api/dma-buf.html) from the original on 1 June 2020. Retrieved 11 June 2020.

142. "About mac80211" (https://wireless.wiki.kernel.org/en/developers/Documentation/mac80211). Linux Kernel Organization, Inc. Archived (https://web.archive.org/web/20210201114135/https://wireless.wiki.kernel.org/en/developers/documentation/mac80211) from the original on 1 February 2021. Retrieved 8 June 2014.
143. "Report on ABI changes in the Linux kernel" (http://abi-laboratory.pro/tracker/timeline/linux/). Andrey Ponomarenko's ABI laboratory. 17 March 2016. Archived (https://web.archive.org/web/20160312025113/http://abi-laboratory.pro/tracker/timeline/linux/) from the original on 12 March 2016. Retrieved 16 March 2016.
144. "[PATCH v3 1/2] fork: add clone3 [LWN.net]" (https://lwn.net/ml/linux-kernel/20190604160944.4058-1-christian@brauner.io/). *lwn.net*. Archived (https://web.archive.org/web/20200716232314/https://lwn.net/ml/linux-kernel/20190604160944.4058-1-christian@brauner.io/) from the original on 16 July 2020. Retrieved 16 July 2020.
145. "clone(2) - Linux manual page" (https://man7.org/linux/man-pages/man2/clone.2.html). *man7.org*. Archived (https://web.archive.org/web/20200715175357/https://man7.org/linux/man-pages/man2/clone.2.html) from the original on 15 July 2020. Retrieved 15 July 2020.
146. "clone3(), fchmodat4(), and fsinfo() [LWN.net]" (https://lwn.net/Articles/792628/). *lwn.net*. Archived (https://web.archive.org/web/20200615080341/https://lwn.net/Articles/792628/) from the original on 15 June 2020. Retrieved 15 July 2020.
147. "ld-linux.so(8) - Linux manual page" (https://man7.org/linux/man-pages/man8/ld-linux.so.8.html). *man7.org*. Archived (https://web.archive.org/web/20201126063027/https://man7.org/linux/man-pages/man8/ld-linux.so.8.html) from the original on 26 November 2020. Retrieved 18 November 2020.
148. "nptl(7) - Linux manual page" (https://man7.org/linux/man-pages/man7/nptl.7.html). *man7.org*. Archived (https://web.archive.org/web/20200725045335/https://man7.org/linux/man-pages/man7/nptl.7.html) from the original on 25 July 2020. Retrieved 25 July 2020.
149. "pthreads(7) - Linux manual page" (https://man7.org/linux/man-pages/man7/pthreads.7.html). *man7.org*. Archived (https://web.archive.org/web/20200715200134/https://man7.org/linux/man-pages/man7/pthreads.7.html) from the original on 15 July 2020. Retrieved 25 July 2020.
150. "pthread_create(3) - Linux manual page" (https://man7.org/linux/man-pages/man3/pthread_create.3.html). *man7.org*. Archived (https://web.archive.org/web/20200725052820/https://man7.org/linux/man-pages/man3/pthread_create.3.html) from the original on 25 July 2020. Retrieved 25 July 2020.
151. "futex(7) - Linux manual page" (https://man7.org/linux/man-pages/man7/futex.7.html). *man7.org*. Archived (https://web.archive.org/web/20200715175424/https://man7.org/linux/man-pages/man7/futex.7.html) from the original on 15 July 2020. Retrieved 25 July 2020.
152. "Kernel threads made easy [LWN.net]" (https://lwn.net/Articles/65178/). *lwn.net*. Archived (https://web.archive.org/web/20200331215714/https://lwn.net/Articles/65178/) from the original on 31 March 2020. Retrieved 15 August 2020.
153. "execve(2) - Linux manual page" (https://www.man7.org/linux/man-pages/man2/execve.2.html). *www.man7.org*. Archived (https://web.archive.org/web/20200715183742/https://man7.org/linux/man-pages/man2/execve.2.html) from the original on 15 July 2020. Retrieved 17 July 2020.
154. "capabilities(7) - Linux manual page" (https://man7.org/linux/man-pages/man7/capabilities.7.html). *man7.org*. Archived (https://web.archive.org/web/20200715201514/https://man7.org/linux/man-pages/man7/capabilities.7.html) from the original on 15 July 2020. Retrieved 2 August 2020.
155. Bar, Moshe (1 April 2000). "The Linux Scheduler" (http://www.linuxjournal.com/article/3910). *Linux Journal*. Belltown Media, Inc. Archived (https://web.archive.org/web/20210202131440/https://www.linuxjournal.com/article/3910) from the original on 2 February 2021. Retrieved 14 April 2012.

156. *BKK19-TR03 - The Linux Kernel Scheduler - Overview* (https://www.youtube.com/watch?v=oOiaRHC9ZDg), retrieved 17 May 2021
157. Love, Robert (2010). "4". *Linux kernel development*. Addison-Wesley. pp. 46–47. ISBN 978-0-672-32946-3. OCLC 268788260 (https://www.worldcat.org/oclc/268788260).
158. Love, Robert (2010). "4". *Linux Kernel Development* (3rd ed.). Addison Wesley. pp. 62–63. ISBN 9780672329463.
159. "Lowering Latency in Linux: Introducing a Preemptible Kernel | Linux Journal" (https://www.linuxjournal.com/article/5600). *www.linuxjournal.com*. Archived (https://web.archive.org/web/20200809182228/https://www.linuxjournal.com/article/5600) from the original on 9 August 2020. Retrieved 17 August 2020.
160. Love, Robert (2010). "4". *Linux Kernel Development* (3rd ed.). Addison Wesley. pp. 46–50. ISBN 9780672329463.
161. "CFS Scheduler — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/scheduler/sched-design-CFS.html#). *www.kernel.org*. Retrieved 1 May 2021.
162. "IEEE Standard for Information Technology – Portable Operating System Interface, POSIX.1b, Real-time extensions (IEEE Std 1003.1b-1993)" (http://www.opengroup.org/onlinepubs/009695399). Archived (https://web.archive.org/web/20101116144926/http://www.opengroup.org/onlinepubs/009695399/) from the original on 16 November 2010. Retrieved 17 March 2016.
163. Larabel, Michael (24 January 2014). "The Linux 3.14 Kernel Already Has Many Exciting Features" (https://www.phoronix.com/scan.php?page=news_item&px=MTU4Mjg). Phoronix. Archived (https://web.archive.org/web/20200813143115/https://www.phoronix.com/scan.php?page=news_item&px=MTU4Mjg) from the original on 13 August 2020. Retrieved 3 February 2014.
164. "Linux kernel 3.14, Section 1.1. Deadline scheduling class for better real-time scheduling" (http://kernelnewbies.org/Linux_3.14#head-651929cdcf19cc2e2cfc7feb16b78ef963d195fe). *kernelnewbies.org*. 30 March 2014. Archived (https://web.archive.org/web/20210115101454/https://kernelnewbies.org/Linux_3.14#head-651929cdcf19cc2e2cfc7feb16b78ef963d195fe) from the original on 15 January 2021. Retrieved 2 April 2014.
165. McKenney, Paul (10 August 2005). "A realtime preemption overview" (https://lwn.net/Articles/146861/). LWN.net. Archived (https://web.archive.org/web/20200810165635/https://lwn.net/Articles/146861/) from the original on 10 August 2020. Retrieved 5 February 2012.
166. "OSADL Project: Realtime Linux" (https://www.osadl.org/Realtime-Linux.projects-realtime-linux.0.html). OSADL. Archived (https://web.archive.org/web/20210204170950/https://www.osadl.org/Realtime-Linux.projects-realtime-linux.0.html) from the original on 4 February 2021. Retrieved 5 February 2012.
167. Love, Robert (2010). "9". *Linux Kernel Development* (3rd ed.). Addison Wesley. p. 167. ISBN 9780672329463.
168. Love, Robert (2010). "10". *Linux Kernel Development* (3rd ed.). Addison Wesley. pp. 176–198. ISBN 9780672329463.
169. "locking.rst - Documentation/kernel-hacking/locking.rst - Linux source code (v5.11.10) - Bootlin" (https://elixir.bootlin.com/linux/v5.11.10/source/Documentation/kernel-hacking/locking.rst). *elixir.bootlin.com*. Retrieved 29 March 2021.
170. "What is RCU, Fundamentally? [LWN.net]" (https://lwn.net/Articles/262464/). *lwn.net*. Retrieved 29 March 2021.
171. "What is RCU? Part 2: Usage [LWN.net]" (https://lwn.net/Articles/263130/). *lwn.net*. Retrieved 29 March 2021.
172. "RCU part 3: the RCU API [LWN.net]" (https://lwn.net/Articles/264090/). *lwn.net*. Retrieved 29 March 2021.
173. "Linux-Kernel Memory Model" (http://open-std.org/JTC1/SC22/WG21/docs/papers/2020/p0124r7.html). *open-std.org*. Retrieved 29 March 2021.

174. "A formal kernel memory-ordering model (part 1) [LWN.net]" (https://lwn.net/Articles/718628/). *lwn.net*. Retrieved 29 March 2021.
175. "A formal kernel memory-ordering model (part 2) [LWN.net]" (https://lwn.net/Articles/720550/). *lwn.net*. Retrieved 29 March 2021.
176. Stern, Alan. "Explanation of the Linux-Kernel Memory Consistency Model" (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/tools/memory-model/Documentation/explanation.txt).
177. Love, Robert (2010). *Linux Kernel Development*. Addison Wesley. pp. 133–137. ISBN 9780672329463.
178. Love, Robert (2010). *Linux Kernel Development*. Addison Wesley. p. 20. ISBN 9780672329463.
179. "The Linux Storage Stack Diagram" (https://www.thomas-krenn.com/de/wikiDE/images/7/72/Linux-storage-stack-diagram_v4.10.svg). *www.thomas-krenn.com*. Archived (https://web.archive.org/web/20200803100605/https://www.thomas-krenn.com/de/wikiDE/images/7/72/Linux-storage-stack-diagram_v4.10.svg) from the original on 3 August 2020. Retrieved 19 March 2020.
180. Torvalds, Linus (January 1999). "The Linux Edge" (https://archive.org/details/isbn_9781565925823). *Open Sources: Voices from the Open Source Revolution*. O'Reilly. ISBN 1-56592-582-3. Retrieved 13 October 2013.
181. "Porting Linux to the DEC Alpha: The Kernel and Shell" (https://www.linuxjournal.com/article/1178?page=0,1). Archived (https://web.archive.org/web/20190905215158/https://www.linuxjournal.com/article/1178?page=0,1) from the original on 5 September 2019. Retrieved 5 October 2019.
182. "Linux on Alpha: A Strategic Choice" (https://www.linuxjournal.com/article/1150?page=0,0). Archived (https://web.archive.org/web/20190904234429/https://www.linuxjournal.com/article/1150?page=0,0) from the original on 4 September 2019. Retrieved 5 October 2019.
183. "Avalon Cluster | TOP500 Supercomputer Sites" (https://www.top500.org/system/166763). *www.top500.org*. Archived (https://web.archive.org/web/20191005210605/https://www.top500.org/system/166763) from the original on 5 October 2019. Retrieved 5 October 2019.
184. Wang, David (6 May 2010). "Android Now Running On iPhone 3G" (https://www.pcworld.com/article/195789/android_now_running_on_iphone_3g.html). *TechHive*. IDG. Archived (https://web.archive.org/web/20100722023655/http://www.pcworld.com/article/195789/android_now_running_on_iphone_3g.html) from the original on 22 July 2010. Retrieved 11 July 2010.
185. "LKDDb" (https://cateee.net/lkddb/). LKDDb Project. Archived (https://web.archive.org/web/20210225020934/https://cateee.net/lkddb/) from the original on 25 February 2021. Retrieved 26 January 2021.
186. "Linux Hardware" (https://linux-hardware.org/). Linux Hardware Project. Archived (https://web.archive.org/web/20210126054431/https://linux-hardware.org/) from the original on 26 January 2021. Retrieved 26 January 2021.
187. "Linux kernel 4.0, Section 1.2. Live patching" (http://kernelnewbies.org/Linux_4.0#head-9aa7c8499b42911a48c02b24f367bf2bc6db8606). *kernelnewbies.org*. 26 April 2015. Archived (https://web.archive.org/web/20150504015114/http://kernelnewbies.org/Linux_4.0#head-9aa7c8499b42911a48c02b24f367bf2bc6db8606) from the original on 4 May 2015. Retrieved 27 April 2015.
188. Jonathan Corbet (25 February 2015). "A rough patch for live patching" (https://lwn.net/Articles/634649/). LWN.net. Archived (https://web.archive.org/web/20150427181127/https://lwn.net/Articles/634649/) from the original on 27 April 2015. Retrieved 27 April 2015.

189. "kernel/git/torvalds/linux.git: Pull live patching infrastructure from Jiri Kosina (Linux kernel source tree)" (https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=1d9c5d79e6e4385aea6f69c23ba543717434ed70). *kernel.org*. 11 February 2015. Archived (https://web.archive.org/web/20150611040359/https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=1d9c5d79e6e4385aea6f69c23ba543717434ed70) from the original on 11 June 2015. Retrieved 27 April 2015.
190. Mookhey, K. K.; Burghate, Nilesh (1 July 2005). *Linux: Security, Audit and Control Features* (https://books.google.com/books?id=-kD0sxQ0EkIC&pg=PA14). USA: ISACA. p. 14. ISBN 1-893209-78-4. Archived (https://web.archive.org/web/20130602223234/http://books.google.com/books?id=-kD0sxQ0EkIC&pg=PA14) from the original on 2 June 2013. Retrieved 31 December 2010.
191. Hatch, Brian (15 July 2008). *Hacking Exposed Linux: Linux Security Secrets and Solutions* (https://books.google.com/books?id=f5Vz08spzw8C&pg=PA524). McGraw-Hill Osborne Media. p. 524. ISBN 978-0-07-226257-5. Archived (https://web.archive.org/web/20130602212901/http://books.google.com/books?id=f5Vz08spzw8C&pg=PA524) from the original on 2 June 2013. Retrieved 31 December 2010.
192. Jaeger, Trent (7 October 2008). *Operating System Security* (https://books.google.com/books?id=P4PYPSv8nBMC&pg=PA122). Morgan and Claypool Publishers. p. 122. ISBN 978-1-59829-212-1. Archived (https://web.archive.org/web/20130602203613/http://books.google.com/books?id=P4PYPSv8nBMC&pg=PA122) from the original on 2 June 2013. Retrieved 31 December 2010.
193. "CAP_PERFMON — and new capabilities in general [LWN.net]" (https://lwn.net/Articles/812719/). *lwn.net*. Archived (https://web.archive.org/web/20200804030704/https://lwn.net/Articles/812719/) from the original on 4 August 2020. Retrieved 2 August 2020.
194. "Linux Security Module Usage — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/admin-guide/LSM/index.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200502142406/https://www.kernel.org/doc/html/latest/admin-guide/LSM/index.html) from the original on 2 May 2020. Retrieved 10 January 2020.
195. "National Security Agency | Central Security Service > What We Do > Research > SE Linux > SE Linux FAQs" (https://www.nsa.gov/What-We-Do/Research/SELinux/FAQs/). *www.nsa.gov*. Archived (https://web.archive.org/web/20190918022139/https://www.nsa.gov/What-We-Do/Research/SELinux/FAQs/) from the original on 18 September 2019. Retrieved 10 January 2020.
196. "Seccomp BPF (SECure COMPuting with filters) — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/userspace-api/seccomp_filter.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200307065527/https://www.kernel.org/doc/html/latest/userspace-api/seccomp_filter.html) from the original on 7 March 2020. Retrieved 10 January 2020.
197. Andrews, Jeremy (16 July 2008). "Security Bugs and Full Disclosure" (https://web.archive.org/web/20080719130436/http://kerneltrap.org/Linux/Security_Bugs_and_Full_Disclosure). KernelTrap. Archived from the original (http://kerneltrap.org/Linux/Security_Bugs_and_Full_Disclosure) on 19 July 2008. Retrieved 31 December 2010.
198. Spengler, Brad (16 July 2008). "Linux's unofficial security-through-coverup policy" (http://seclists.org/fulldisclosure/2008/Jul/276). *Full Disclosure* (Mailing list). Archived (https://web.archive.org/web/20200807161645/https://seclists.org/fulldisclosure/2008/Jul/276) from the original on 7 August 2020. Retrieved 31 December 2010.
199. Foundation, The Linux (25 October 2017). "2017 State of Linux Kernel Development" (https://www.linuxfoundation.org/publications/2017/10/2017-state-of-linux-kernel-development/). *The Linux Foundation*. Archived (https://web.archive.org/web/20200527074644/https://www.linuxfoundation.org/publications/2017/10/2017-state-of-linux-kernel-development/) from the original on 27 May 2020. Retrieved 27 May 2020.

200. "git-clone(1) - Linux manual page" (https://man7.org/linux/man-pages/man1/git-clone.1.html). *man7.org*. Archived (https://web.archive.org/web/20201014211233/https://man7.org/linux/man-pages/man1/git-clone.1.html) from the original on 14 October 2020. Retrieved 16 August 2020.
201. "git-pull(1) - Linux manual page" (https://man7.org/linux/man-pages/man1/git-pull.1.html). *man7.org*. Archived (https://web.archive.org/web/20201112020325/https://man7.org/linux/man-pages/man1/git-pull.1.html) from the original on 12 November 2020. Retrieved 16 August 2020.
202. Robert Love (2010). *Linux Kernel Development: Linux Kernel Development*. Pearson Education. p. 11. ISBN 9780768696790.
203. Robert Love (2010). *Linux Kernel Development: Linux Kernel Development*. Pearson Education. p. 12. ISBN 9780768696790.
204. "How the development process works" (https://www.kernel.org/doc/html/latest/process/2.Process.html). Archived (https://web.archive.org/web/20171209130758/https://www.kernel.org/doc/html/latest/process/2.Process.html) from the original on 9 December 2017. Retrieved 4 February 2018.
205. Robert Love (2010). *Linux Kernel Development: Linux Kernel Development*. Pearson Education. p. 13. ISBN 9780768696790.
206. "HOWTO do Linux kernel development — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/process/howto.html#documentation). *www.kernel.org*. Archived (https://web.archive.org/web/20200307065439/https://www.kernel.org/doc/html/latest/process/howto.html#documentation) from the original on 7 March 2020. Retrieved 4 January 2020.
207. "Linux kernel coding style — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/process/coding-style.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200105083545/https://www.kernel.org/doc/html/latest/process/coding-style.html) from the original on 5 January 2020. Retrieved 4 January 2020.
208. Kubbilun, Ingo A. (2 June 2004). "Linux kernel patch for Intel Compiler" (https://web.archive.org/web/20110722090031/http://www.pyrillion.org/index.html?showframe=linuxkernelpatch.html) (in German). Pyrillion.org. Archived from the original (http://www.pyrillion.org/index.html?showframe=linuxkernelpatch.html) on 22 July 2011. Retrieved 12 November 2010.
209. timothy (26 February 2009). "High Performance Linux Kernel Project — LinuxDNA" (http://linux.slashdot.org/article.pl?sid=09/02/26/2216241). *Slashdot Linux*. Dice Holdings. Archived (https://web.archive.org/web/20191018044639/https://linux.slashdot.org/story/09/02/26/2216241/high-performance-linux-kernel-project-linuxdna) from the original on 18 October 2019. Retrieved 30 October 2010.
210. Ryan, Justin (25 February 2009). "LinuxDNA Supercharges Linux with the Intel C/C++ Compiler" (http://www.linuxjournal.com/content/linuxdna-supercharges-linux-intel-cc-compiler). *Linux Journal*. Belltown Media, Inc. Archived (https://web.archive.org/web/20201109011614/https://www.linuxjournal.com/content/linuxdna-supercharges-linux-intel-cc-compiler) from the original on 9 November 2020. Retrieved 30 October 2010.
211. Lelbach, Bryce (25 October 2010). "Clang builds a working Linux Kernel (Boots to RL5 with SMP, networking and X, self hosts)" (https://web.archive.org/web/20150907044958/http://lists.cs.uiuc.edu/pipermail/cfe-dev/2010-October/011711.html). *cfe-dev* (Mailing list). Archived from the original (http://lists.cs.uiuc.edu/pipermail/cfe-dev/2010-October/011711.html) on 7 September 2015.
212. Larabel, Michael (12 April 2014). "Linux 3.15 Can Almost Be Compiled Under LLVM's Clang" (https://www.phoronix.com/scan.php?page=news_item&px=MTY2MjY). Phoronix. Archived (https://web.archive.org/web/20200813143201/https://www.phoronix.com/scan.php?page=news_item&px=MTY2MjY) from the original on 13 August 2020. Retrieved 10 June 2014.

213. Larabel, Michael. "Patch By Patch, LLVM Clang Gets Better At Building The Linux Kernel" (https://www.phoronix.com/scan.php?page=news_item&px=MTY2MjY). Phoronix. Archived (https://web.archive.org/web/20200813143201/https://www.phoronix.com/scan.php?page=news_item&px=MTY2MjY) from the original on 13 August 2020. Retrieved 20 November 2014.
214. Edge, Jake (7 May 2013). "LFCS: The LLVMLinux project" (https://lwn.net/Articles/549203/). LWN.net. Archived (https://web.archive.org/web/20200810165632/https://lwn.net/Articles/549203/) from the original on 10 August 2020. Retrieved 3 March 2015.
215. Möller, Jan-Simon (2 February 2014). "LLVMLinux: The Linux Kernel with Dragon Wings" (http://llvm.org/devmtg/2014-02/slides/moller-llvmlinux.pdf) (PDF). LLVM Project. Archived (https://web.archive.org/web/20200803053328/http://llvm.org/devmtg/2014-02/slides/moller-llvmlinux.pdf) (PDF) from the original on 3 August 2020. Retrieved 3 March 2015.
216. Desaulniers, Nick; Hackmann, Greg; Hines, Stephen (18 October 2017). "2017 LLVM Developers' Meeting: Compiling Android userspace and Linux kernel with LLVM" (https://www.youtube.com/watch?v=6l4DtR5exwo&t=2130). YouTube. Archived (https://web.archive.org/web/20201231030548/https://www.youtube.com/watch?v=6l4DtR5exwo&t=2130) from the original on 31 December 2020. Retrieved 7 December 2020.
217. Hackmann, Greg (2 February 2017). "marlin-nougat-mr1-clang Patch Series" (https://android-review.googlesource.com/q/topic:marlin-nougat-mr1-clang+(status:open+OR+status:closed)). Archived (https://web.archive.org/web/20201210125624/https://android-review.googlesource.com/q/topic:marlin-nougat-mr1-clang+(status:open+OR+status:closed)) from the original on 10 December 2020. Retrieved 6 December 2020.
218. Kaehlcke, Matthias (22 October 2018). "cros-kernel2: Make clang the default compiler for kernel builds" (https://chromium-review.googlesource.com/c/chromiumos/overlays/chromiumos-overlay/+/1294370). Archived (https://web.archive.org/web/20201210015343/https://chromium-review.googlesource.com/c/chromiumos/overlays/chromiumos-overlay/+/1294370) from the original on 10 December 2020. Retrieved 6 December 2020.
219. Larabel, Michael (4 February 2019). "Using LLVM Clang To Compile The Linux Kernel Is Heating Up Again Thanks To Google" (https://www.phoronix.com/scan.php?page=news_item&px=Google-2019-Clang-Kernel). Phoronix. Archived (https://web.archive.org/web/20201125201932/https://www.phoronix.com/scan.php?page=news_item&px=Google-2019-Clang-Kernel) from the original on 25 November 2020. Retrieved 6 December 2020.
220. Desaulniers, Nick (10 December 2019). "vts: kernel: enforce vts_kernel_toolchain for all TARGET_ARCH for R" (https://android-review.googlesource.com/c/platform/test/vts-testcase/kernel/+/1185200). Archived (https://web.archive.org/web/20201210125711/https://android-review.googlesource.com/c/platform/test/vts-testcase/kernel/+/1185200) from the original on 10 December 2020. Retrieved 6 December 2020.
221. Desaulniers, Nick (19 November 2020). "Re: violating function pointer signature" (https://lore.kernel.org/lkml/CAKwvOdmKjsJGbR7hHACk3qUgguy-HWvJQerwTnArE0AwhPgfcQ@mail.gmail.com/). LKML. Retrieved 6 December 2020.
222. Love, Robert (2010). *Linux kernel development* (3rd ed.). Addison-Wesley. p. 364. ISBN 978-0-672-32946-3. OCLC 268788260 (https://www.worldcat.org/oclc/268788260).
223. Bradford, John (8 March 2003). "Re: what's an OOPS" (http://lkml.iu.edu/hypermail/linux/kernel/0303.1/0009.html). *LKML* (Mailing list). Archived (https://web.archive.org/web/20141031032356/http://lkml.iu.edu/hypermail/linux/kernel/0303.1/0009.html) from the original on 31 October 2014. Retrieved 30 October 2010.
224. Love, Robert (2010). *Linux kernel development*. Addison Wesley. p. 371. ISBN 9780672329463. OCLC 268788260 (https://www.worldcat.org/oclc/268788260).
225. "syslog(2) - Linux manual page" (https://man7.org/linux/man-pages/man2/syslog.2.html). *man7.org*. Archived (https://web.archive.org/web/20201013152012/https://man7.org/linux/man-pages/man2/syslog.2.html) from the original on 13 October 2020. Retrieved 15 August 2020.

226. "kmsg: export printk records to the /dev/kmsg interface [LWN.net]" (https://lwn.net/Articles/493182/). *lwn.net*. Archived (https://web.archive.org/web/20151002050933/http://lwn.net/Articles/493182/) from the original on 2 October 2015. Retrieved 16 August 2020.
227. "systemd" (https://www.freedesktop.org/wiki/Software/systemd/). *www.freedesktop.org*. Archived (https://web.archive.org/web/20200818000645/https://www.freedesktop.org/wiki/Software/systemd/) from the original on 18 August 2020. Retrieved 16 August 2020.
228. "systemd-journald(8) - Linux manual page" (https://man7.org/linux/man-pages/man8/systemd-journald.8.html). *man7.org*. Archived (https://web.archive.org/web/20200812003040/https://www.man7.org/linux/man-pages/man8/systemd-journald.8.html) from the original on 12 August 2020. Retrieved 15 August 2020.
229. *See what your computer is doing with Ftrace utilities* (https://www.youtube.com/watch?v=68osT1soAPM), retrieved 9 May 2021
230. "Debugging the kernel using Ftrace - part 1 [LWN.net]" (https://lwn.net/Articles/365835/). *lwn.net*. Archived (https://web.archive.org/web/20201109001219/https://lwn.net/Articles/365835/) from the original on 9 November 2020. Retrieved 15 September 2020.
231. "Debugging the kernel using Ftrace - part 2 [LWN.net]" (https://lwn.net/Articles/366796/). *lwn.net*. Archived (https://web.archive.org/web/20200331222229/https://lwn.net/Articles/366796/) from the original on 31 March 2020. Retrieved 15 September 2020.
232. "ftrace - Function Tracer — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/trace/ftrace.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200919095357/https://www.kernel.org/doc/html/latest/trace/ftrace.html) from the original on 19 September 2020. Retrieved 15 September 2020.
233. "Boot-time tracing — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/trace/boottime-trace.html). *www.kernel.org*. Archived (https://web.archive.org/web/20201031200922/https://www.kernel.org/doc/html/latest/trace/boottime-trace.html) from the original on 31 October 2020. Retrieved 19 September 2020.
234. "Kernel Probes (Kprobes) — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/trace/kprobes.html). *www.kernel.org*. Archived (https://web.archive.org/web/20201011030448/https://www.kernel.org/doc/html/latest/trace/kprobes.html) from the original on 11 October 2020. Retrieved 6 October 2020.
235. "Uprobe-tracer: Uprobe-based Event Tracing — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/trace/uprobetracer.html). *www.kernel.org*. Archived (https://web.archive.org/web/20201204204113/https://www.kernel.org/doc/html/latest/trace/uprobetracer.html) from the original on 4 December 2020. Retrieved 6 October 2020.
236. "Using kgdb, kdb and the kernel debugger internals" (https://mirrors.edge.kernel.org/pub/linux/kernel/people/jwessel/kdb/index.html). *mirrors.edge.kernel.org*. Archived (https://web.archive.org/web/20210126003430/https://mirrors.edge.kernel.org/pub/linux/kernel/people/jwessel/kdb/index.html) from the original on 26 January 2021. Retrieved 3 November 2020.
237. Gene Sally (2010). *Pro Linux Embedded Systems*. Apress. p. 252. ISBN 9781430272267.
238. "Code of Conflict" (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/process/code-of-conflict.rst). Retrieved 4 February 2018.
239. Sharwood, Simon (6 October 2015). "Linux kernel dev who asked Linus Torvalds to stop verbal abuse quits over verbal abuse" (https://www.theregister.co.uk/2015/10/06/linix_kernel_dev_who_asked_linus_torvalds_to_stop_swearing_quits_over_swearing/). *The Register*. Archived (https://web.archive.org/web/20200329075939/https://www.theregister.co.uk/2015/10/06/linix_kernel_dev_who_asked_linus_torvalds_to_stop_swearing_quits_over_swearing/) from the original on 29 March 2020. Retrieved 4 February 2018.
240. Edge, Jake (31 January 2018). "Too many lords, not enough stewards" (https://lwn.net/Articles/745817/). *LWN.net*. Archived (https://web.archive.org/web/20201109004145/https://lwn.net/Articles/745817/) from the original on 9 November 2020. Retrieved 4 February 2018.

241. Corbet, Jonathan (6 November 2017). "Bash the kernel maintainers" (https://lwn.net/Articles/738222/). *LWN.net*. Archived (https://web.archive.org/web/20210126003428/https://lwn.net/Articles/738222/) from the original on 26 January 2021. Retrieved 4 February 2018.
242. Billimoria, Kaiwan N. (2021). *Linux Kernel Programming A Comprehensive Guide to Kernel Internals, Writing Kernel Modules, and Kernel Synchronization* (https://www.worldcat.org/oclc/1240585605). Birmingham: Packt Publishing, Limited. p. 55. ISBN 978-1-78995-592-7. OCLC 1240585605 (https://www.worldcat.org/oclc/1240585605).
243. Vaduva, Alexandru (2016). *Linux : embedded development : leverage the power of Linux to develop captivating and powerful embedded Linux projects : a course in three modules* (https://www.worldcat.org/oclc/960471438). Alex Gonzalez, Chris Simmonds. Birmingham, UK. p. 663. ISBN 978-1-78712-445-5. OCLC 960471438 (https://www.worldcat.org/oclc/960471438).
244. *Building embedded Linux systems* (https://www.worldcat.org/oclc/273049576). Karim Yaghmour (2nd ed.). Sebastopol [Calif.]: O'Reilly Media. 2008. p. 387. ISBN 978-0-596-52968-0. OCLC 273049576 (https://www.worldcat.org/oclc/273049576).
245. Yaghmour, Karim (2011). *Embedded Android* (https://www.worldcat.org/oclc/812180000). Sebastopol, CA: O'Reilly Media. p. 44. ISBN 978-1-4493-2798-9. OCLC 812180000 (https://www.worldcat.org/oclc/812180000).
246. "SoC (System on a Chip)" (https://openwrt.org/docs/techref/hardware/soc). *OpenWrt Wiki*. 6 November 2014. Retrieved 15 March 2021.
247. "What to do about CVE numbers [LWN.net]" (https://lwn.net/Articles/801157/). *lwn.net*. Retrieved 15 March 2021.
248. Amadeo, Ron (20 November 2019). "Google outlines plans for mainline Linux kernel support in Android" (https://arstechnica.com/gadgets/2019/11/google-outlines-plans-for-mainline-linux-kernel-support-in-android/). *Ars Technica*. Retrieved 31 March 2021.
249. Bruchon, Jody (24 April 2021), *jbruchon/elks* (https://github.com/jbruchon/elks), retrieved 27 April 2021
250. "The state of preempt-rt" (https://web.archive.org/web/20161015044835/https://linuxplumbersconf.org/ocw/proposals/49). *linuxplumbersconf.org*. Archived from the original (https://linuxplumbersconf.org/ocw/proposals/49) on 15 October 2016. Retrieved 14 June 2016.
251. Meyer, David (3 February 2010). "Linux developer explains Android kernel code removal" (https://www.zdnet.com/article/linux-developer-explains-android-kernel-code-removal/). *ZDNet*. CBS Interactive. Archived (https://web.archive.org/web/20161015045139/http://www.zdnet.com/article/linux-developer-explains-android-kernel-code-removal/) from the original on 15 October 2016. Retrieved 3 February 2010.
252. "Chapter 03: maemo Platform Overview" (https://web.archive.org/web/20080616191310/http://maemo.org/maemo_training_material/maemo4.x/html/maemo_Technology_Overview/Chapter_03_maemo_Platform_Overview.html). *maemo Technology Overview*. Nokia. 2008. Archived from the original (http://maemo.org/maemo_training_material/maemo4.x/html/maemo_Technology_Overview/Chapter_03_maemo_Platform_Overview.html) on 16 June 2008. Retrieved 9 April 2010.
253. Kroah-Hartman, Greg (2 February 2010). "Android and the Linux kernel community" (http://www.kroah.com/log/linux/android-kernel-problems.html). Archived (https://web.archive.org/web/20190427144039/http://www.kroah.com/log/linux/android-kernel-problems.html) from the original on 27 April 2019. Retrieved 3 February 2010.
254. Roger Ye (2017). *Android System Programming*. Packt Publishing. p. 14. ISBN 9781787120389.
255. Torvalds, Linus (19 September 2001). "The Origins of Linux—Linus Torvalds" (https://www.youtube.com/watch?v=WVTWCPoUt8w&t=3435). YouTube. Retrieved 6 December 2020.

256. "Why I quit: kernel developer Con Kolivas" (https://web.archive.org/web/20110707151924/http://apcmag.com/why_i_quit_kernel_developer_con_kolivas.htm). *APC Magazine*. ACP Magazines. 24 July 2007. Archived from the original (http://apcmag.com/node/6735/) on 7 July 2011. Retrieved 15 August 2011.
257. Corbet, Jonathan (25 July 2007). "Re: -mm merge plans for 2.6.23" (https://lwn.net/Articles/242768/). LWN.net. Archived (https://web.archive.org/web/20180211131406/https://lwn.net/Articles/242768/) from the original on 11 February 2018. Retrieved 10 February 2018.
258. Cox, Alan (28 July 2009). "Re: [PATCH] kdesu broken" (https://lkml.org/lkml/2009/7/28/375). Archived (https://web.archive.org/web/20180211190040/https://lkml.org/lkml/2009/7/28/375) from the original on 11 February 2018. Retrieved 10 February 2018.
259. Rodrigues, Goldwyn (22 January 2011). "A tale of two SCSI targets" (https://lwn.net/Articles/424004/). Archived (https://web.archive.org/web/20180215204201/https://lwn.net/Articles/424004/) from the original on 15 February 2018. Retrieved 14 February 2018.
260. Steinmetz, Andreas (17 January 2013). "LIO - the broken iSCSI target implementation" (https://lkml.org/lkml/2013/1/16/803). Archived (https://web.archive.org/web/20180215204140/https://lkml.org/lkml/2013/1/16/803) from the original on 15 February 2018. Retrieved 14 February 2018.
261. Paul, Ryan (19 June 2012). "Linus Torvalds says "f–k you" to NVIDIA" (https://arstechnica.com/information-technology/2012/06/linus-torvalds-says-f-k-you-to-nvidia/). Archived (https://web.archive.org/web/20180215023959/https://arstechnica.com/information-technology/2012/06/linus-torvalds-says-f-k-you-to-nvidia/) from the original on 15 February 2018. Retrieved 14 February 2018.
262. John Gold (3 April 2014). "Linus Torvalds suspends key Linux developer: Kernel panic as Systemd dev pokes the bear" (https://www.networkworld.com/article/2175826/linus-torvalds-suspends-key-linux-developer.html). Archived (https://web.archive.org/web/20190324195212/https://www.networkworld.com/article/2175826/linus-torvalds-suspends-key-linux-developer.html) from the original on 24 March 2019. Retrieved 24 March 2019.
263. Poettering, Lennart (6 October 2014). "On the sickness of the Linux Kernel Community" (https://web.archive.org/web/20180527195108/https://plus.google.com/+LennartPoetteringTheOneAndOnly/posts/J2TZrTvu7vd). *Google+*. Archived from the original (https://plus.google.com/+LennartPoetteringTheOneAndOnly/posts/J2TZrTvu7vd) on 27 May 2018. Retrieved 10 February 2018.
264. Brodkin, Jon (6 March 2015). "VMware alleged to have violated Linux's open source license for years" (https://arstechnica.com/tech-policy/2015/03/vmware-alleged-to-have-violated-linuxs-open-source-license-for-years/). *Ars Technica*. Archived (https://web.archive.org/web/20180215023512/https://arstechnica.com/tech-policy/2015/03/vmware-alleged-to-have-violated-linuxs-open-source-license-for-years/) from the original on 15 February 2018. Retrieved 14 February 2018.
265. McCarthy, Kieren (26 August 2016). "Having offended everyone else in the world, Linus Torvalds calls own lawyers a 'nasty festering disease' " (https://www.theregister.co.uk/2016/08/26/linus_torvalds_calls_own_lawyers_nasty_festering_disease/). *The Register*. Archived (https://web.archive.org/web/20180215023540/https://www.theregister.co.uk/2016/08/26/linus_torvalds_calls_own_lawyers_nasty_festering_disease/) from the original on 15 February 2018. Retrieved 14 February 2018.
266. Corbet, Jonathan (10 September 2007). "KS2007: Developer relations and development process" (https://lwn.net/Articles/249104/). *LWN.net*. Archived (https://web.archive.org/web/20180212142109/https://lwn.net/Articles/249104/) from the original on 12 February 2018. Retrieved 11 February 2018.

267. Brodkin, Jon (16 July 2013). "Linus Torvalds defends his right to shame Linux kernel developers" (https://arstechnica.com/information-technology/2013/07/linus-torvalds-defends-his-right-to-shame-linux-kernel-developers/). *ARS Technica*. Archived (https://web.archive.org/web/20180217143017/https://arstechnica.com/information-technology/2013/07/linus-torvalds-defends-his-right-to-shame-linux-kernel-developers/) from the original on 17 February 2018. Retrieved 11 February 2018.
268. Corbet, Jonathan (9 March 2015). "The kernel's code of conflict" (https://lwn.net/Articles/635999/). *LWN.net*. Archived (https://web.archive.org/web/20180212142143/https://lwn.net/Articles/635999/) from the original on 12 February 2018. Retrieved 11 February 2018.
269. Corbet, Jonathan (18 September 2018). "Code, conflict, and conduct" (https://lwn.net/SubscriberLink/765108/f1a80a6d6a6ff0f4/). LWN.net. Archived (https://web.archive.org/web/20180919175320/https://lwn.net/SubscriberLink/765108/f1a80a6d6a6ff0f4/) from the original on 19 September 2018. Retrieved 19 September 2018.
270. Cohen, Noam (19 September 2018). "After Years of Abusive E-mails, the Creator of Linux Steps Aside" (https://www.newyorker.com/science/elements/after-years-of-abusive-e-mails-the-creator-of-linux-steps-aside). *The New Yorker*. Archived (https://web.archive.org/web/20200220085413/https://www.newyorker.com/science/elements/after-years-of-abusive-e-mails-the-creator-of-linux-steps-aside) from the original on 20 February 2020. Retrieved 24 September 2018.
271. Larabel, Michael. "Dropping Profanity In Kernel Code Comments: Linux Gets "Hugs" " (https://www.phoronix.com/scan.php?page=news_item&px=Linux-Kernel-Hugs). *Phoronix*. Archived (https://web.archive.org/web/20190421094724/https://www.phoronix.com/scan.php?page=news_item&px=Linux-Kernel-Hugs) from the original on 21 April 2019. Retrieved 15 June 2019.
272. "Linux Evolution" (http://www.sprg.uniroma2.it/kernelhacking2008/lectures/lkhc08-01b.pdf) (PDF). 26 March 2008. Archived (https://web.archive.org/web/20131214074153/http://www.sprg.uniroma2.it/kernelhacking2008/lectures/lkhc08-01b.pdf) (PDF) from the original on 14 December 2013. Retrieved 6 November 2013.
273. "Perpetual Development: A Model of the Linux Kernel Life Cycle" (http://www.cs.huji.ac.il/~feit/papers/LinuxDev12JSS.pdf) (PDF). 25 October 2011. Archived (https://web.archive.org/web/20131017210855/http://www.cs.huji.ac.il/~feit/papers/LinuxDev12JSS.pdf) (PDF) from the original on 17 October 2013. Retrieved 6 November 2013.
274. Kroah-Hartman, Greg (12 February 2008). "Re: Announce: Linux-next (Or Andrew's dream :-))" (http://lkml.iu.edu/hypermail/linux/kernel/0802.1/2159.html). *Linux Kernel Mailing List* (Mailing list). Archived (https://web.archive.org/web/20170202070946/http://lkml.iu.edu/hypermail/linux/kernel/0802.1/2159.html) from the original on 2 February 2017. Retrieved 30 January 2017.
275. Wheeler, David A. "Linux Kernel 2.6: It's Worth More!" (http://www.dwheeler.com/essays/linux-kernel-cost.html). Archived (https://www.webcitation.org/616XYjg7d?url=http://www.dwheeler.com/essays/linux-kernel-cost.html) from the original on 21 August 2011. Retrieved 18 January 2007.
276. "Economic impact of FLOSS on innovation and competitiveness of the EU ICT sector" (http://ec.europa.eu/enterprise/sectors/ict/files/2006-11-20-flossimpact_en.pdf) (PDF) (Table 3 on page 50). Archived (https://web.archive.org/web/20100215190539/http://ec.europa.eu/enterprise/sectors/ict/files/2006-11-20-flossimpact_en.pdf) (PDF) from the original on 15 February 2010. Retrieved 8 January 2011.
277. "Estimating Total Development Cost Of a Linux Distribution" (https://web.archive.org/web/20100711025812/http://www.linuxfoundation.org/publications/estimatinglinux.pdf) (PDF) (Table on page 6). Archived from the original (http://www.linuxfoundation.org/publications/estimatinglinux.pdf) (PDF) on 11 July 2010.

278. "The Billion Dollar Kernel" (http://linux.slashdot.org/story/10/02/24/155214/The-Billion-Dollar-Kernel). Linux.slashdot.org. 24 February 2010. Archived (https://web.archive.org/web/20110515003125/http://linux.slashdot.org/story/10/02/24/155214/The-Billion-Dollar-Kernel) from the original on 15 May 2011. Retrieved 12 November 2010.
279. Wheeler, David. "The Linux Kernel: It's Worth More!" (https://dwheeler.com/essays/linux-kernel-cost.html). Archived (https://web.archive.org/web/20210224011056/https://dwheeler.com/essays/linux-kernel-cost.html) from the original on 23 February 2021. Retrieved 17 September 2012.
280. "Linux MAINTAINERS file" (https://archive.today/20130112231112/http://git.kernel.org/?p=linux/kernel/git/torvalds/linux.git;a=blob;f=MAINTAINERS). Archived from the original (https://git.kernel.org/?p=linux/kernel/git/torvalds/linux.git;a=blob;f=MAINTAINERS) on 12 January 2013.
281. Torvalds, Linus (16 September 2018). "Linux 4.19-rc4 released, an apology, and a maintainership note" (https://lkml.org/lkml/2018/9/16/167). *LKML*. Archived (https://web.archive.org/web/20180923085327/https://lkml.org/lkml/2018/9/16/167) from the original on 23 September 2018. Retrieved 23 September 2018.
282. Alexandru Vaduva, Alex Gonzalez & Chris Simmonds (2016). *Linux: Embedded Development*. Packt Publishing. p. 663. ISBN 9781787124455.
283. "The Linux Kernel Archives" (https://www.kernel.org/). Archived (https://web.archive.org/web/20110221140221/http://www.kernel.org/) from the original on 21 February 2011. Retrieved 13 November 2019.
284. Yamagata, Hiroo (3 August 1997). "The Pragmatist of Free Software" (https://web.archive.org/web/20070210224351/http://hotwired.goo.ne.jp/matrix/9709/5_linus.html). HotWired. Archived from the original (http://hotwired.goo.ne.jp/matrix/9709/5_linus.html) on 10 February 2007. Retrieved 21 February 2007.
285. "GPL-v2" (https://www.gnu.org/licenses/old-licenses/gpl-2.0.html). *gnu.org*. Archived (https://web.archive.org/web/20191225033729/https://www.gnu.org/licenses/old-licenses/gpl-2.0.html) from the original on 25 December 2019. Retrieved 28 January 2020.
286. Corbet, Jonathan (31 January 2006). "GPLv3 and the kernel" (https://lwn.net/Articles/169797/). LWN.net. Archived (https://web.archive.org/web/20200810165701/https://lwn.net/Articles/169797/) from the original on 10 August 2020. Retrieved 21 February 2007.
287. Torvalds, Linus (8 September 2000). "Linux-2.4.0-test8" (http://lkml.iu.edu/hypermail/linux/kernel/0009.1/0096.html). *LKML* (Mailing list). Archived (https://web.archive.org/web/20200515235654/http://lkml.iu.edu/hypermail/linux/kernel/0009.1/0096.html) from the original on 15 May 2020. Retrieved 21 February 2007.
288. "gnu.org" (https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html#section9). *www.gnu.org*. Archived (https://web.archive.org/web/20210202151435/https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html#section9) from the original on 2 February 2021. Retrieved 18 October 2017.
289. Cox, Alan (20 January 2006). "Re: GPL V3 and Linux" (https://lwn.net/Articles/169831/). *LKML* (Mailing list). Archived (https://web.archive.org/web/20210126131909/https://lwn.net/Articles/169831/) from the original on 26 January 2021. Retrieved 21 February 2007.
290. Shankland, Stephen (25 September 2006). "Top Linux programmers pan GPL 3" (http://news.com/Top+Linux+programmers+pan+GPL+3/2100-7344_3-6119372.html). *News.com*. CNET. Retrieved 21 February 2007.
291. James E.J. Bottomley, Mauro Carvalho Chehab, Thomas Gleixner, Christoph Hellwig, Dave Jones, Greg Kroah-Hartman, Tony Luck, Andrew Morton, Trond Myklebust, David Woodhouse (15 September 2006). "Kernel developers' position on GPLv3: The Dangers and Problems with GPLv3" (https://lwn.net/Articles/200422/). LWN.net. Archived (https://web.archive.org/web/20210118015213/https://lwn.net/Articles/200422/) from the original on 18 January 2021. Retrieved 11 March 2015.

292. Petreley, Nicholas (27 September 2006). "A fight against evil or a fight for attention?" (http://www.linuxjournal.com/node/1000100). linuxjournal.com. Archived (https://web.archive.org/web/20180302144635/http://www.linuxjournal.com/node/1000100) from the original on 2 March 2018. Retrieved 11 March 2015.
293. "Linus Torvalds says GPL v3 violates everything that GPLv2 stood for" (https://www.youtube.com/watch?v=PaKIZ7gJlRU). Debconf 2014. 2014. Archived (https://web.archive.org/web/20180508034417/https://www.youtube.com/watch?v=PaKIZ7gJlRU) from the original on 8 May 2018. Retrieved 21 March 2018.
294. Clark, Rob; Semwal, Sumit (1 November 2012). "DMA Buffer Sharing Framework: An Introduction" (http://elinux.org/images/a/a8/DMA_Buffer_Sharing-_An_Introduction.pdf) (PDF). Embedded Linux Conference. Archived (https://web.archive.org/web/20140808051804/http://elinux.org/images/a/a8/DMA_Buffer_Sharing-_An_Introduction.pdf) (PDF) from the original on 8 August 2014. Retrieved 2 August 2014.
295. Cox, Alan (10 October 2012). "[PATCH] dma-buf: Use EXPORT_SYMBOL" (http://lists.freedesktop.org/archives/dri-devel/2012-October/028846.html). *Direct Rendering Infrastructure* (Mailing list). Archived (https://web.archive.org/web/20130122222858/http://lists.freedesktop.org/archives/dri-devel/2012-October/028846.html) from the original on 22 January 2013. Retrieved 3 September 2013.
296. Torvalds, Linus (10 December 2003). "RE: Linux GPL and binary module exception clause?" (https://lkml.org/lkml/2003/12/10/123). *LKML* (Mailing list). Archived (https://web.archive.org/web/20110615102501/http://lkml.org/lkml/2003/12/10/123) from the original on 15 June 2011. Retrieved 31 December 2010.
297. Torvalds, Linus (3 December 2003). "Re: Linux GPL and binary module exception clause?" (http://lkml.iu.edu/hypermail/linux/kernel/0312.0/0670.html). *LKML* (Mailing list). Archived (https://web.archive.org/web/20200428052533/http://lkml.iu.edu/hypermail/linux/kernel/0312.0/0670.html) from the original on 28 April 2020. Retrieved 12 November 2010.
298. "Linux Firmware API — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/driver-api/firmware/index.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200113174720/https://www.kernel.org/doc/html/latest/driver-api/firmware/index.html) from the original on 13 January 2020. Retrieved 13 January 2020.
299. "Tainted kernels — The Linux Kernel documentation" (https://www.kernel.org/doc/html/latest/admin-guide/tainted-kernels.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200307065211/https://www.kernel.org/doc/html/latest/admin-guide/tainted-kernels.html) from the original on 7 March 2020. Retrieved 13 January 2020.
300. "Built-in firmware — The Linux Kernel documentation" (https://www.kernel.org/doc/html/v4.16/driver-api/firmware/built-in-fw.html). *www.kernel.org*. Archived (https://web.archive.org/web/20200610041327/https://www.kernel.org/doc/html/v4.16/driver-api/firmware/built-in-fw.html) from the original on 10 June 2020. Retrieved 10 June 2020.
301. "Linux TM registration in the US" (http://tmsearch.uspto.gov/bin/showfield?f=doc&state=4808:r0ouik.2.17). *uspto.gov*. Archived (https://web.archive.org/web/20210224164104/http://tmsearch.uspto.gov/bin/showfield?f=doc&state=4808:r0ouik.2.17) from the original on 24 February 2021. Retrieved 6 September 2019.
302. "Linux TM registration in the EU" (https://euipo.europa.eu/eSearch/#details/trademarks/000851246). *euipo.europa.eu*. Archived (https://wayback.archive-it.org/all/20160609153529/https://euipo.europa.eu/eSearch/#details/trademarks/000851246) from the original on 9 June 2016. Retrieved 28 November 2020.
303. Hughes, Phil (1 August 1997). "Linux Trademark Dispute" (http://www.linuxjournal.com/article/2425/). *Linux Journal*. Belltown Media, Inc. Archived (https://web.archive.org/web/20100430060209/http://www.linuxjournal.com/article/2425) from the original on 30 April 2010. Retrieved 8 December 2010.

304. Hughes, Phil (1 March 1997). "Action Taken on Linux Trademark" (http://www.linuxjournal.com/article/2098). *Linux Journal*. Belltown Media, Inc. Archived (https://web.archive.org/web/20100303180921/http://www.linuxjournal.com/article/2098) from the original on 3 March 2010. Retrieved 8 December 2010.
305. Gisselberg, Tonya (2010). "The Trademark History of Linux, the Operating System" (https://web.archive.org/web/20110711095344/http://www.gisselberglawfirm.com/downloads/linux.pdf) (PDF). Gisselberg Law Firm, Inc. Archived from the original (http://www.gisselberglawfirm.com/downloads/linux.pdf) (PDF) on 11 July 2011. Retrieved 8 December 2010.

Further reading

- Torvalds, Linus; Diamond, David (2001). *Just for Fun: The Story of an Accidental Revolutionary*. HarperBusiness. ISBN 978-0066620732.
- Bezroukov, Nikolai. "Ch 4: A benevolent dictator" (http://www.softpanorama.org/People/Torvalds/index.shtml). *Portraits of Open Source Pioneers* (e-book). Softpanorama. Archived (https://web.archive.org/web/20051013082354/http://www.softpanorama.org/People/Torvalds/index.shtml) from the original on 13 October 2005. Retrieved 3 October 2005.
- "LinkSys and binary modules" (https://lwn.net/Articles/53780/). LWN.net Weekly Edition. 16 October 2003. Archived (https://web.archive.org/web/20160801080742/http://lwn.net/Articles/53780/) from the original on 1 August 2016. Retrieved 21 July 2016.
- "Everyone's Favorite Linux Mascot" (http://www.nd.edu/~ljordan/linux/tuxhistory.html). Archived (https://web.archive.org/web/20050816235544/http://www.nd.edu/~ljordan/linux/tuxhistory.html) from the original on 16 August 2005. Retrieved 16 June 2005.
- Pranevich, Joseph (December 2003). "The Wonderful World of Linux 2.6" (https://web.archive.org/web/20030716054145/http://www.kniggit.net/wwol26.html). Archived from the original (http://kniggit.net/wwol26.html) on 16 July 2003.
- "LinuxChanges" (http://wiki.kernelnewbies.org/LinuxChanges). Archived (https://web.archive.org/web/20051031211753/http://wiki.kernelnewbies.org/LinuxChanges) from the original on 31 October 2005. Retrieved 31 October 2005.
- "Seminar Paper on Linux Kernel 2.6" (https://web.archive.org/web/20070202002917/http://www.engineeringproject.net/seminars/linux.htm). Archived from the original (http://engineeringproject.net/seminars/linux.htm) on 2 February 2007.
- "Linux Device Drivers" (https://lwn.net/Kernel/LDD3/) (3rd ed.). Archived (https://web.archive.org/web/20160727085953/http://lwn.net/Kernel/LDD3/) from the original on 27 July 2016. Retrieved 21 July 2016.
- "Understanding the Linux Kernel" (http://www.oreilly.com/catalog/understandlk/) (Book) (3rd ed.). Archived (https://web.archive.org/web/20051217094234/http://www.oreilly.com/catalog/understandlk/) from the original on 17 December 2005. Retrieved 22 December 2005.
- *Linux Kernel Networking, by Rami Rosen, 2014* (http://www.apress.com/9781430261964) (Book). Archived (https://web.archive.org/web/20150512052750/http://www.apress.com/9781430261964) from the original on 12 May 2015. Retrieved 14 June 2015.
- "Linux: The GPL And Binary Modules" (https://web.archive.org/web/20050723031159/http://kerneltrap.org/node/1735). Archived from the original (http://kerneltrap.org/node/1735) on 23 July 2005.
- "Anatomy of the Linux kernel" (http://www.ibm.com/developerworks/linux/library/l-linux-kernel/). Archived (https://web.archive.org/web/20070627093507/http://www.ibm.com/developerworks/linux/library/l-linux-kernel/) from the original on 27 June 2007. Retrieved 9 June 2007.

External links

- Official website (https://kernel.org)

    - Linux kernel documentation index (https://kernel.org/doc/)
    - Linux kernel man pages (https://kernel.org/doc/man-pages/)
    - Kernel bugzilla (https://bugzilla.kernel.org/), and regressions (https://bugzilla.kernel.org/show_bug.cgi?id=15790) for each recent kernel version
- Kernel Newbies (http://kernelnewbies.org/), a source of various kernel-related information
- Kernel coverage at LWN.net (https://lwn.net/Kernel/), an authoritative source of kernel-related information
- Bootlin's Elixir Cross Referencer (https://elixir.bootlin.com/linux/latest/source), a Linux kernel source code cross-reference
- Greg Kroah Hartman on the Linux kernel (https://www.youtube.com/watch?v=L2SED6sewRw) on YouTube