**Practical no-10**

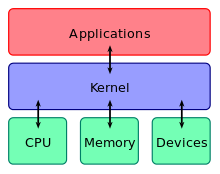
Linux Kernel:

Learn linux kernel with respect to:

1. **What is linux kernel:-** The Linux kernel is an operating system (OS) kernel defined as Unix-like in nature. It used in different operating systems, mostly in the form of different Linux distributions.

The Linux kernel was the first truly complete and prominent example of free and open-source software that prompted its wide adoption and received contributions from thousands of developers.

1. **Operating model:-** The **kernel** is a [computer program](https://en.wikipedia.org/wiki/Computer_program) that is the core of a computer's [operating system](https://en.wikipedia.org/wiki/Operating_system), with complete control over everything in the system.[[1]](https://en.wikipedia.org/wiki/Kernel_(operating_system)#cite_note-Linfo-1) On most systems, it is one of the first programs loaded on [start-up](https://en.wikipedia.org/wiki/Booting) (after the [bootloader](https://en.wikipedia.org/wiki/Bootloader" \o "Bootloader)). It handles the rest of start-up as well as [input/output](https://en.wikipedia.org/wiki/Input/output) requests from [software](https://en.wikipedia.org/wiki/Software), translating them into [data-processing](https://en.wikipedia.org/wiki/Data_processing) instructions for the [central processing unit](https://en.wikipedia.org/wiki/Central_processing_unit). It handles memory and [peripherals](https://en.wikipedia.org/wiki/Peripheral) like keyboards, monitors, printers, and speakers.

[](https://en.wikipedia.org/wiki/File:Kernel_Layout.svg)

A kernel connects the [application software](https://en.wikipedia.org/wiki/Application_software) to the hardware of a computer.

The critical code of the kernel is usually loaded into a protected area of memory, which prevents it from being overwritten by [applications](https://en.wikipedia.org/wiki/Application_software) or other, more minor parts of the operating system. The kernel performs its tasks, such as running processes and handling interrupts, in [kernel space](https://en.wikipedia.org/wiki/Kernel_space). In contrast, everything a user does is in [user space](https://en.wikipedia.org/wiki/User_space): writing text in a text editor, running programs in a [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface), etc. This separation prevents user data and kernel data from interfering with each other and causing instability and slowness

1. **Licensing model:-** Torvalds released Linux under a license which forbade any commercial use.[[36]](https://en.wikipedia.org/wiki/Linux_kernel#cite_note-hiroo-36) This was changed in version 0.12 by a switch to the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License) (GPL).[[20]](https://en.wikipedia.org/wiki/Linux_kernel#cite_note-Relnotes-0.12-20)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.

Torvalds has described licensing Linux under the GPL as the "best thing I ever did.

]

The Linux kernel is licensed explicitly only under version 2 of the GPL, without offering the licensee the option to choose "any later version", which is a common GPL extension. 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. Torvalds himself specifically indicated upon the release of version 2.4.0 that his own code is released only under version 2.[[38]](https://en.wikipedia.org/wiki/Linux_kernel#cite_note-38) However, the terms of the GPL state that if no version is specified, then any version may be used,[[*verification needed*](https://en.wikipedia.org/wiki/Wikipedia:Verifiability)] and [Alan Cox](https://en.wikipedia.org/wiki/Alan_Cox) pinted out that very few other Linux contributors had specified a particular version of the GPL.

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."[[40]](https://en.wikipedia.org/wiki/Linux_kernel#cite_note-40) This group of high-profile kernel developers, including [Linus Torvalds](https://en.wikipedia.org/wiki/Linus_Torvalds), [Greg Kroah-Hartman](https://en.wikipedia.org/wiki/Greg_Kroah-Hartman) and [Andrew Morton](https://en.wikipedia.org/wiki/Andrew_Morton_(computer_programmer)), commented on mass media about their objections to the GPLv3.[[41]](https://en.wikipedia.org/wiki/Linux_kernel#cite_note-kerneldevelopers2006-41) They referred to clauses regarding [DRM](https://en.wikipedia.org/wiki/Digital_rights_management)/[tivoization](https://en.wikipedia.org/wiki/Tivoization), patents, "additional restrictions" and warned a [Balkanisation](https://en.wikipedia.org/wiki/Balkanisation) of the "Open Source Universe" by the GPLv3.[[41]](https://en.wikipedia.org/wiki/Linux_kernel#cite_note-kerneldevelopers2006-41)[[42]](https://en.wikipedia.org/wiki/Linux_kernel#cite_note-42) Linus Torvalds, who decided not to adopt the GPLv3 for the Linux kernel, reiterated his criticism even years later.

1. **How development works:-**

The kernel developers use a loosely time-based release process, with a new major kernel release happening every two or three months. The recent release history looks like this:

|  |  |
| --- | --- |
| 2.6.38 | March 14, 2011 |
| 2.6.37 | January 4, 2011 |
| 2.6.36 | October 20, 2010 |
| 2.6.35 | August 1, 2010 |
| 2.6.34 | May 15, 2010 |
| 2.6.33 | February 24, 2010 |

Every 2.6.x release is a major kernel release with new features, internal API changes, and more. A typical 2.6 release can contain nearly 10,000 changesets with changes to several hundred thousand lines of code. 2.6 is thus the leading edge of Linux kernel development; the kernel uses a rolling development model which is continually integrating major changes.

A relatively straightforward discipline is followed with regard to the merging of patches for each release. At the beginning of each development cycle, the “merge window” is said to be open. At that time, code which is deemed to be sufficiently stable (and which is accepted by the development community) is merged into the mainline kernel. The bulk of changes for a new development cycle (and all of the major changes) will be merged during this time, at a rate approaching 1,000 changes (“patches,” or “changesets”) per day.

(As an aside, it is worth noting that the changes integrated during the merge window do not come out of thin air; they have been collected, tested, and staged ahead of time. How that process works will be described in detail later on).

The merge window lasts for approximately two weeks. At the end of this time, Linus Torvalds will declare that the window is closed and release the first of the “rc” kernels. For the kernel which is destined to be 2.6.40, for example, the release which happens at the end of the merge window will be called 2.6.40-rc1. The -rc1 release is the signal that the time to merge new features has passed, and that the time to stabilize the next kernel has begun.

Over the next six to ten weeks, only patches which fix problems should be submitted to the mainline. On occasion a more significant change will be allowed, but such occasions are rare; developers who try to merge new features outside of the merge window tend to get an unfriendly reception. As a general rule, if you miss the merge window for a given feature, the best thing to do is to wait for the next development cycle. (An occasional exception is made for drivers for previously-unsupported hardware; if they touch no in-tree code, they cannot cause regressions and should be safe to add at any time).

As fixes make their way into the mainline, the patch rate will slow over time. Linus releases new -rc kernels about once a week; a normal series will get up to somewhere between -rc6 and -rc9 before the kernel is considered to be sufficiently stable and the final 2.6.x release is made. At that point the whole process starts over again.