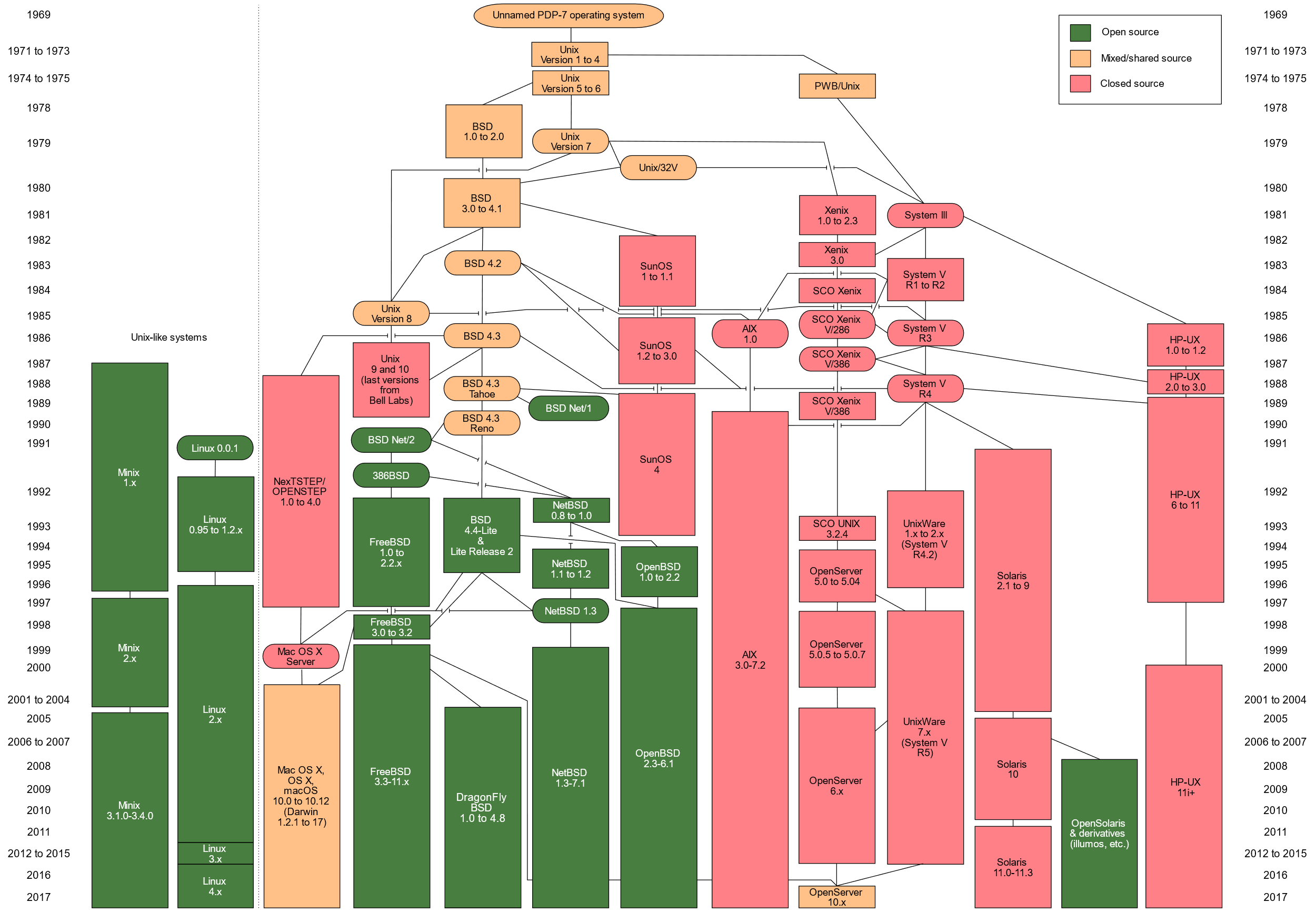
**DevOps**

**Day-1**

BSD - Berkeley Software Distribution 🡪 UNIX OS based derivative developed by CSRG of University of California, Berkeley. BSD has been considered a branch of Unix, **Berkeley Unix**, because it shared the initial codebase and design with the original [AT&T](https://en.wikipedia.org/wiki/AT%26T) Unix operating system. In the 1980s, BSD was widely adopted by vendors of [workstation](https://en.wikipedia.org/wiki/Computer_workstation)-class systems in the form of proprietary Unix variants such as [DEC](https://en.wikipedia.org/wiki/Digital_Equipment_Corporation) [ULTRIX](https://en.wikipedia.org/wiki/ULTRIX) and [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems) [SunOS](https://en.wikipedia.org/wiki/SunOS). This can be attributed to the ease with which it could be licensed, and the familiarity the founders of many technology companies of the time had with it. BSD releases provided a basis for several [open source](https://en.wikipedia.org/wiki/Open_source) development projects that are ongoing, including [FreeBSD](https://en.wikipedia.org/wiki/FreeBSD), [OpenBSD](https://en.wikipedia.org/wiki/OpenBSD), [NetBSD](https://en.wikipedia.org/wiki/NetBSD), [DragonFlyBSD](https://en.wikipedia.org/wiki/DragonFlyBSD" \o "DragonFlyBSD), [Darwin,](https://en.wikipedia.org/wiki/Darwin_(operating_system)) and [TrueOS](https://en.wikipedia.org/wiki/TrueOS" \o "TrueOS).



Evolution of UNIX

UNIX - is a family of [multitasking](https://en.wikipedia.org/wiki/Computer_multitasking), [multiuser](https://en.wikipedia.org/wiki/Multiuser) computer [operating systems](https://en.wikipedia.org/wiki/Operating_system) that derive from the original [AT&T](https://en.wikipedia.org/wiki/AT%26T_Corporation) Unix. Unix was originally meant to be a convenient platform for programmers developing software to be run on it and on other systems, rather than for non-programmer users. The system grew larger as the operating system started spreading in academic circles, as users added their own tools to the system and shared them with colleagues. Unix systems are characterized by various concepts: the use of plain text for storing data; a hierarchical file system; treating devices and certain types of inter-process communication (IPC) as files; and the use of a large number of software tools, small programs that can be strung together through a command-line interpreter using pipes, as opposed to using a single monolithic program that includes all of the same functionality. These concepts are collectively known as the "Unix philosophy".

The Unix system is composed of several components that were originally packaged together. By including the development environment, libraries, documents and the portable, modifiable source code for all of these components, in addition to the kernel of an operating system, Unix was a self-contained software system. This was one of the key reasons it emerged as an important teaching and learning tool and has had such a broad influence.

Initially intended for use inside the [Bell System](https://en.wikipedia.org/wiki/Bell_System), AT&T licensed Unix to outside parties from the late 1970s, leading to a variety of both academic and commercial variants of Unix from vendors such as the [University of California, Berkeley](https://en.wikipedia.org/wiki/University_of_California,_Berkeley) ([BSD](https://en.wikipedia.org/wiki/Berkeley_Software_Distribution)), [Microsoft](https://en.wikipedia.org/wiki/Microsoft) ([Xenix](https://en.wikipedia.org/wiki/Xenix" \o "Xenix)), [IBM](https://en.wikipedia.org/wiki/IBM) ([AIX](https://en.wikipedia.org/wiki/AIX)) and [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems) ([Solaris](https://en.wikipedia.org/wiki/Solaris_(operating_system))). AT&T finally sold its rights in Unix to [Novell](https://en.wikipedia.org/wiki/Novell) in the early 1990s, which then sold its Unix business to the [Santa Cruz Operation](https://en.wikipedia.org/wiki/Santa_Cruz_Operation) (SCO) in 1995,[[4]](https://en.wikipedia.org/wiki/Unix#cite_note-4) but the UNIX trademark passed to the industry standards consortium [The Open Group](https://en.wikipedia.org/wiki/The_Open_Group), which allows the use of the mark for certified operating systems compliant with the [Single UNIX Specification](https://en.wikipedia.org/wiki/Single_UNIX_Specification) (SUS). Among these is [Apple](https://en.wikipedia.org/wiki/Apple_Inc.)'s [macOS](https://en.wikipedia.org/wiki/MacOS),[[5]](https://en.wikipedia.org/wiki/Unix#cite_note-5) which is the Unix version with the largest installed base as of 2014.

LINUX - Linux is a name that broadly denotes a family of free and open-source software operating systems built around the Linux kernel. Typically, Linux is packaged in a form known as a Linux distribution for both desktop and server use. The defining component of a Linux distribution is the Linux kernel, an operating system kernel first released on September 17, 1991, by Linus Torvalds.

In 1991, while attending the [University of Helsinki](https://en.wikipedia.org/wiki/University_of_Helsinki), Torvalds became curious about operating systems.[[39]](https://en.wikipedia.org/wiki/Linux#cite_note-41) Frustrated by the licensing of MINIX, which at the time limited it to educational use only,[[38]](https://en.wikipedia.org/wiki/Linux#cite_note-minix-lic-40) he began to work on his own operating system kernel, which eventually became the [Linux kernel](https://en.wikipedia.org/wiki/Linux_kernel).Torvalds began the development of the Linux kernel on MINIX and applications written for MINIX were also used on Linux. Later, Linux matured and further Linux kernel development took place on Linux systems.[[40]](https://en.wikipedia.org/wiki/Linux#cite_note-42) GNU applications also replaced all MINIX components, because it was advantageous to use the freely available code from the GNU Project with the fledgling operating system; code licensed under the GNU GPL can be reused in other computer programs as long as they also are released under the same or a compatible license. Torvalds initiated a switch from his original license, which prohibited commercial redistribution, to the GNU GPL.[[41]](https://en.wikipedia.org/wiki/Linux#cite_note-43) Developers worked to integrate GNU components with the Linux kernel, making a fully functional and free operating system.[[42]](https://en.wikipedia.org/wiki/Linux#cite_note-gnu_history-44)

A Linux-based system is a modular Unix-like operating system, deriving much of its basic design from principles established in Unix during the 1970s and 1980s. Such a system uses a monolithic kernel, the Linux kernel, which handles process control, networking, access to the peripherals, and file systems. Device drivers are either integrated directly with the kernel, or added as modules that are loaded while the system is running.

The primary difference between Linux and many other popular contemporary operating systems is that the Linux kernel and other components are free and open-source software. Linux is not the only such operating system, although it is by far the most widely used. Some free and open-source software licenses are based on the principle of copyleft, a kind of reciprocity: any work derived from a copyleft piece of software must also be copyleft itself. The most common free software license, the GNU General Public License (GPL), is a form of copyleft, and is used for the Linux kernel and many of the components from the GNU Project.

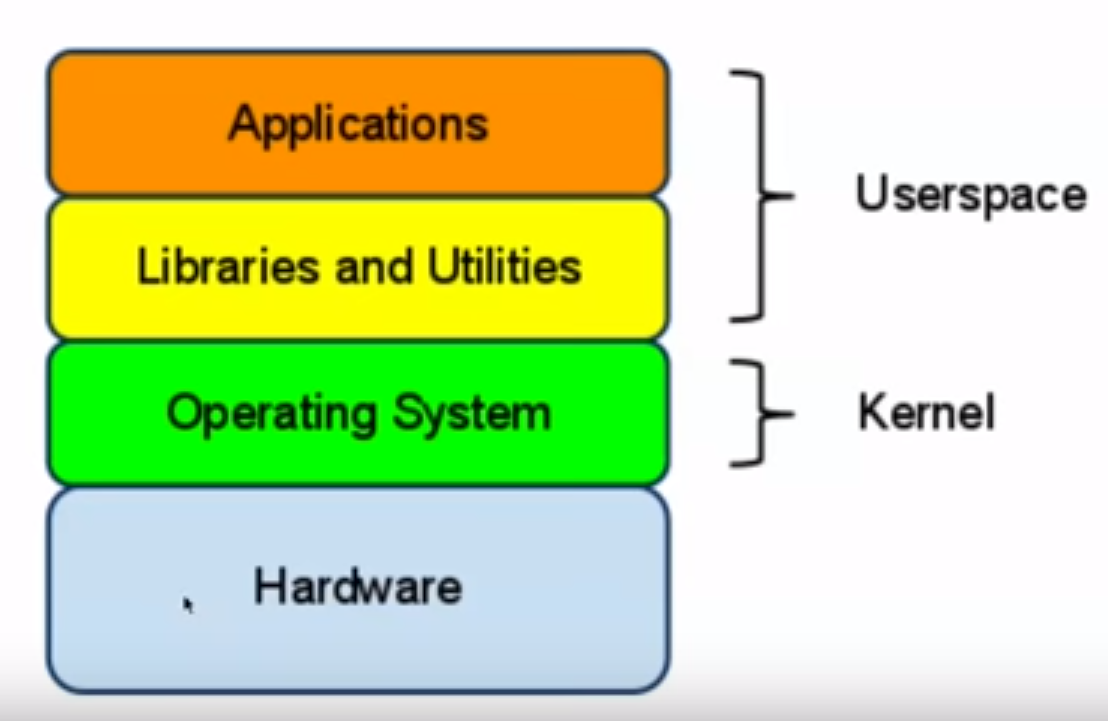
OPERATING SYSTEM - An Operating System is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs. It also allows you to **communicate** with the computer without knowing how to speak the computer's language.  Most of the time, there are several different computer programs running at the same time, and they all need to access your computer's **central processing unit (CPU)**, **memory**, and **storage**. The operating system coordinates all of this to make sure each program gets what it needs.



Examples for OS are Microsoft Windows, Apple Macintosh, Linux, centOS, REDHAT, Ubuntu etc. Examples for Mobile OS include Apple IOS, Android, Symbian, Blackberry OS etc.

Following are some of important functions of an operating System.

* Memory Management
* Processor Management
* Device Management
* File Management
* Security
* Control over system performance
* Job accounting
* Error detecting aids
* Coordination between other software and users



The components of an operating system all exist in order to make the different parts of a computer work together. All user software needs to go through the operating system in order to use any of the hardware, whether it be as simple as a mouse or keyboard or as complex as an Internet component.

* **Kernel:**  The Kernel provides the most basic level of control over all of the computer's hardware devices. It manages memory access for programs in the [RAM](https://en.wikipedia.org/wiki/Random-access_memory), it determines which programs get access to which hardware resources, it sets up or resets the CPU's operating states for optimal operation at all times.
* **Program Execution:** The operating system is also a set of services which simplify development and execution of application programs. Executing an application program involves the creation of a process by the operating system [kernel](https://en.wikipedia.org/wiki/Kernel_(computer_science)) which assigns memory space and other resources, establishes a priority for the process in multi-tasking systems, loads program binary code into memory, and initiates execution of the application program which then interacts with the user and with hardware devices.
* **Interrupt:** When an interrupt is received, the computer's hardware automatically suspends whatever program is currently running, saves its status, and runs computer code previously associated with the interrupt; this is analogous to placing a bookmark in a book in response to a phone call. In modern operating systems, interrupts are handled by the operating system's [kernel](https://en.wikipedia.org/wiki/Kernel_(computer_science)). Interrupts may come from either the computer's hardware or the running program.
* **Modes:** CPUs with this capability offer at least two modes: [user mode](https://en.wikipedia.org/wiki/User_mode) and [supervisor mode](https://en.wikipedia.org/wiki/Supervisor_mode). In general terms, supervisor mode operation allows unrestricted access to all machine resources, including all MPU instructions. User mode operation sets limits on instruction use and typically disallows direct access to machine resources. CPUs might have other modes similar to user mode as well, such as the virtual modes in order to emulate older processor types, such as 16-bit processors on a 32-bit one, or 32-bit processors on a [64-bit](https://en.wikipedia.org/wiki/64-bit_computing) one.
* **Multitasking**
* **Disk Management & File Systems**
* **Device Drivers**
* **Networking**
* **Security**
* **User Interface**

**Kernel**

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)). 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.

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.

The kernel's [interface](https://en.wikipedia.org/wiki/Application_programming_interface) is a [low-level](https://en.wikipedia.org/wiki/High-_and_low-level) [abstraction layer](https://en.wikipedia.org/wiki/Abstraction_layer). When a [process](https://en.wikipedia.org/wiki/Process_(computing)) makes requests of the kernel, it is called a [system call](https://en.wikipedia.org/wiki/System_call). Kernel designs differ in how they manage these system calls and [resources](https://en.wikipedia.org/wiki/Resource_(computer_science)). A [monolithic kernel](https://en.wikipedia.org/wiki/Monolithic_kernel) runs all the operating system [instructions](https://en.wikipedia.org/wiki/Instruction_set) in the same [address space](https://en.wikipedia.org/wiki/Address_space) for speed. A [microkernel](https://en.wikipedia.org/wiki/Microkernel) runs most processes in user space,[[2]](https://en.wikipedia.org/wiki/Kernel_(operating_system)#cite_note-2) for [modularity](https://en.wikipedia.org/wiki/Modular_programming).

Kernel's primary function is to mediate access to the computer's resources, including

**1.CPU**

**2.RAM**

**3.I/O DEVICES**

**4.Memory Management**

**5.System Calls.**

**Shell**

Simply put, the shell is a program that takes commands from the keyboard and gives them to the operating system to perform. In the old days, it was the only user interface available on a Unix-like system such as Linux. Nowadays, we have *graphical user interfaces (GUIs)* in addition to *command line interfaces (CLIs)*such as the shell.

On most Linux systems a program called [bash](http://linuxcommand.org/lc3_man_pages/bash1.html) (which stands for Bourne Again SHell, an enhanced version of the original Unix shell program, sh, written by Steve Bourne) acts as the shell program. Besides bash, there are other shell programs that can be installed in a Linux system. These include: ksh, tcsh and zsh.

There are various ways to get shell access:

* **Terminal** - Linux desktop provide a GUI based login system. Once logged in you can gain access to a shell by running X Terminal (XTerm), Gnome Terminal (GTerm), or KDE Terminal (KTerm) application.
* **Connect via secure shell (SSH)** - You will get a shell prompt as soon as you log in into remote server or workstation.
* **Use the console** - A few Linux system also provides a text-based login system. Generally you get a shell prompt as soon as you log in to the system

**Day-2**

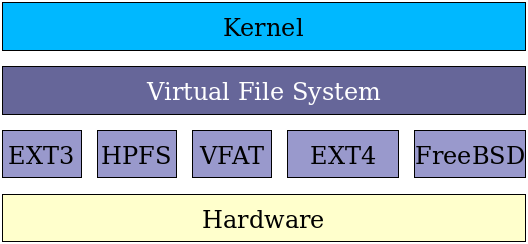
**File System :** “On Linux everything is a File. If something is not a File it is a Process.”

Linux uses a hierarchical file system structure, much like an upside-down tree, with root (/) at the base of the file system and all other directories spreading from there.



|  |  |
| --- | --- |
| **S.No.** | **Directory & Description** |
| 1 | **/**  This is the root directory which should contain only the directories needed at the top level of the file structure |
| 2 | **/bin**  This is where the executable files are located. These files are available to all users |
| 3 | **/sbin –** System Binaries  Just like /bin, /sbin also contains binary executables. But, the linux commands located under this directory are used typically by system aministrator, for system maintenance purpose. For example: iptables, reboot, fdisk, ifconfig, swapon |
| 4 | **/dev**  These are device drivers |
| 5 | **/etc**  Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, ethernet, hosts, where to send critical messages |
| 6 | **/lib**  Contains shared library files and sometimes other kernel-related files. Contains library files that supports the binaries located under /bin and /sbin. |
| 7 | **/boot**  Contains files for booting the system. Kernel initrd, vmlinux, grub files are located under /bootFor example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic. |
| 8 | **/home**  Contains the home directory for users and other accounts |
| 9 | **/mnt**  Used to mount other temporary file systems, such as **cdrom** and **floppy** for the **CD-ROM** drive and **floppy diskette drive**, respectively. |
| 10 | **/proc**  Contains all processes marked as a file by **process number** or other information that is dynamic to the system |
| 11 | **/tmp**  Holds temporary files used between system boots |
| 12 | **/usr**  Used for miscellaneous purposes, and can be used by many users. Includes administrative commands, shared files, library files, and others |
| 13 | **/var**  Typically contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data |
| 14 | **/sbin**  Contains binary (executable) files, usually for system administration. For example, ***fdisk*** and ***ifconfig*** utlities |
| 15 | **/kernel**  Contains kernel files |
| 16 | **/srv –** Service Data  srv stands for service. Contains server specific services related data. For example, /srv/cvs contains CVS related data. |
| 17 | **/media –** Removable Media Devices  Temporary mount directory for removable devices. For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer |
| 17 | **/opt –** Optional add-on Applications  opt stands for optional. Contains add-on applications from individual vendors. Add-on applications should be installed under either /opt/ or /opt/ sub-directory. |

Linux supports multiple types of File Systems which use their own metadata structures to define how the data is stored and accessed.

****

**LINUX COMMANDS**

**Installation of Software in Linux**

The standard Linux package format (according to the Linux Standard Base) is *RPM*. RPM is a packaging system originally developed by Red Hat and widely used in the Linux community. Distributions using it include Fedora, Mandriva, Red Hat (naturally), and SUSE. An RPM package file normally will be named something like program-version-other.rpm

Another popular package format is *DEB*, the Debian software package. Debian packages and the Advanced Packaging Tool (APT) were the first to introduce several advanced features that are now common, such as automatic dependency resolution and signed packages. Debian packages are used by Debian GNU/Linux (naturally), and distributions based on it, including Ubuntu, Knoppix, and Mepis. A Debian package file normally will be named something like program-version-other.deb

Remember, you will need to become SuperUser to install software.

**Debian, Ubuntu: APT**

There is a broad array of tools for working with DEB packages, but the one you will commonly use is apt-get, arguably the easiest of Linux package management tools. apt-getis so easy because it not only keeps track of what packages are installed, but also what other packages are available. It will even download them from the Internet for you (if properly configured).

apt-get install ${packagename}

To remove software is just as easy.

apt-get remove ${packagename}

Although the repositories that contain installable packages might live on the Internet or on a disc somewhere, APT keeps a local database on your hard drive with a list of all available packages and where to find them. This database needs to be explicitly updated. To update the APT database:

apt-get update

## **Fedora, Red Hat: yum**

yum does for RPM packages roughly what apt-get does for Debian packages. Like apt-get, yum can download and install packages from a configured repository.

yum install ${packagename}

To remove software is just as easy.

yum remove ${packagename}

yum does not keep a local copy of your package database by default, so normally there is no need to update it. To install all available security patches and bug fixes, use this command:

yum update

You can also explicitly update a single package with:

yum update ${packagename}

**VI Editor**

The default editor that comes with the UNIX operating system is called vi (**vi**sual editor). There are many ways to edit files in Unix. Editing files using the screen-oriented text editor vi is one of the best ways. This editor enables you to edit lines in context with other lines in the file.

You can use the vi editor to edit an existing file or to create a new file from scratch. You can also use this editor to just read a text file.

**vi's Modes and Moods**

vi has two modes: the command mode and the insert mode. It is essential that you know which mode you are in at any given point in time. When you are in command mode, letters of the keyboard will be interpreted as commands. When you are in insert mode the same letters of the keyboard will type or edit text. vi always starts out in command mode. When you wish to move between the two modes, keep these things in mind. You can type **i** to enter the insert mode. If you wish to leave insert mode and return to the command mode, hit the **ESC** key. If you're not sure where you are, hit **ESC** a couple of times and that should put you back in command mode.

**General Command Information**

As mentioned previously, vi uses letters as commands. It is important to note that in general vi commands:

* are case sensitive - lowercase and uppercase command letters do different things
* are not displayed on the screen when you type them
* generally do not require a **Return** after you type the command.

You will see some commands which start with a colon (**:**). These commands are *ex* commands which are used by the *ex* editor. *ex*is the true editor which lies underneath vi -- in other words, vi is the interface for the ex editor.

**Entering Text**

To begin entering text in an empty file, you must first change from the command mode to the insert mode. To do this, type the letter **i**. When you start typing, anything you type will be entered into the file. Type a few short lines and hit **Return** at the end of each of line.

**Starting the vi Editor**

The following table lists out the basic commands to use the vi editor −

|  |  |
| --- | --- |
| **S.No.** | **Command & Description** |
| 1 | **vi filename**  Creates a new file if it already does not exist, otherwise opens an existing file. |
| 2 | **vi -R filename**  Opens an existing file in the read-only mode. |
| 3 | **view filename**  Opens an existing file in the read-only mode. |

The following points will add to your success with vi −

* You must be in command mode to use the commands. (Press Esc twice at any time to ensure that you are in command mode.)
* You must be careful with the commands. These are case-sensitive.
* You must be in insert mode to enter text.

Wget

**wget** stands for "web get". It is a command-line utility which downloads files over a network. **wget** is a free utility for non-interactive download of files from the web. **wget** is non-interactive, meaning that it can work in the background, while the user is not logged on, which allows you to start a retrieval and disconnect from the system, letting **wget** finish the work. By contrast, most web browsers require constant user interaction, which make transferring a lot of data difficult. **wget** has been designed for robustness over slow or unstable network connections; if a download fails due to a network problem, it will keep retrying until the whole file has been retrieved. If the [server](https://www.computerhope.com/jargon/s/server.htm) supports regetting, it will instruct the server to continue the download from where it left off.

wget http://website.com/files/file.zip

...would download the file into the working directory.

you can install **wget** with [**apt-get**](https://www.computerhope.com/unix/apt-get.htm):

sudo yum install wget

## **wget syntax**

wget [*option*]... [*URL*]...

**Basic Startup Options**

|  |  |
| --- | --- |
| **-V**, **--version** | Display the version of **wget**, and exit. |
| **-h**, **--help** | Print a help message describing all of **wget**'s command-line options, and exit. |
| **-b**, **--background** | Go to background immediately after startup. If no output file is specified via the **-o**, output is redirected to **wget-log** |
| **-e** *command*, **--execute***command* | Execute *command* as if it were a part of the file **.wgetrc**. A command thus invoked will be executed after the commands in **.wgetrc**, thus taking precedence over them. |

**Logging and Input File Options**

|  |  |
| --- | --- |
| **-o** *logfile*, **--output-file=***logfile* | Log all messages to *logfile*. The messages are normally reported to standard error. |
| **-a** *logfile*, **--append-output=***logfile* | Append to *logfile*. This option is the same as **-o**, only it appends to *logfile*instead of overwriting the old log file. If *logfile* does not exist, a new file is created. |
| **-d**, **--debug** | Turn on [debug](https://www.computerhope.com/jargon/d/debug.htm) output, meaning various information important to the developers of **wget** if it does not work properly. Your [system administrator](https://www.computerhope.com/jargon/r/root.htm) may have chosen to compile **wget** without debug support, in which case **-d** will not work.  Note that compiling with debug support is always safe; **wget** compiled with the debug support will not print any debug info unless requested with **-d**. |
| **-q**, **--quiet** | Turn off **wget**'s output. |
| **-v**, **--verbose** | Turn on [verbose](https://www.computerhope.com/jargon/v/verbose.htm) output, with all the available data. The default output is verbose. |
| **-nv**, **--non-verbose** | Non-verbose output. Turn off verbose without being completely quiet (use **-q** for that), which means that error messages and basic information still get printed. |
| **-i** *file*, **--input-file=***file* | Read URLs from a local or external file. If "**-**" is specified as *file*, URLs are read from the [standard input](https://www.computerhope.com/jargon/s/stdin.htm). (Use "**./-**" to read from a file literally named "**-**".)  If this function is used, no URLs need be present on the command line. If there are URLs both on the command line and input file, those on the command lines will be the first ones to be retrieved. If **--force-html** is not specified, then *file* should consist of a series of URLs, one per line. However, if you specify **--force-html**, the document will be regarded as HTML. In that case you may have problems with relative links, which you can solve either by adding **<base href="***url***">** to the documents or by specifying **--base=***url* on the command line.  If the file is an external one, the document will be automatically treated as HTML if the **Content-Type** is "**text/html**". Furthermore, the file's location will be implicitly used as **base href** if none was specified. |
| **-F**, **--force-html** | When input is read from a file, force it to be treated as an HTML file. This enables you to retrieve relative links from existing HTML files on your local disk, by adding **<base href="***url***">** to HTML, or using the **--base command-line** option. |
| **-B** *URL* **--base=***URL* | Resolves relative links using URL as the point of reference, when reading links from an HTML file specified via the **-i**/**--input-file** option (together with **--force-html**, or when the input file was fetched remotely from a server describing it as HTML). This option is equivalent to the presence of a "**BASE**" tag in the HTML input file, with URL as the value for the "**href**" attribute. For instance, if you specify **http://foo/bar/a.html** for URL, and **wget**reads **../baz/b.html** from the input file, it would be resolved to **http://foo/baz/b.html**. |
| **--config=***FILE* | Specify the location of a startup file you want to use. |