The History of Unix

The success of the UNIX system stems from its tasteful selection of a few key ideas and their elegant implementation. The model of the Unix system has led a generation of software designers to new ways of thinking about programming. The genius of the Unix system is its framework, which enables programmers to stand on the work of others. (the Turing Award selection committee, 1983)

Origins

The Unix operating system is a watershed operating system developed at the then AT&T Bell Laboratories by a group of employees including Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy and Joe Ossanna starting in 1969. In the past three decades, the operating system has had tremendous influence on the way we think about and develop operating systems and computer programs in general. Unix as well as various Unix-like and Unix-derived operating systems continue to dominate the computer world even today. In the fast changing world of Information Technology, few computer programs have survived for such a long time while still retaining their essential characteristics.

The people who developed Unix were originally working on a joint project of At&T Bell Laboratories, General Electric (GE) and Massachusetts Institute of Technology (MIT) to develop a large ambitious multi-user operating system called Multics (Multiplexed Information and Computing Service). Though the project had many innovations to its credit, it was a large and unwieldy project that was not progressing as expected. When AT&T decided to pull out of the project, Thomson, Ritchie and others decided to develop a smaller operating system to keep alive some of the ideas of Multics. Ken Thomson had developed a game called Space Travel while still on the Multics projects, but it was too expensive to run the game on a large machine in active use. He found a little used machine at Bell Labs and redeveloped Space Travel to run on it. He and his group gradually added the operating system ideas they had in mind and finally came out with a simple operating system that they initially called Unics, because it supported a single user; as opposed to Multics. When it was developed further and started supporting multiple users, the spelling of the name was changed to Unix.

Ken Thompson and Denis Ritchie

Key Success Factors

Unix was the first successful system to have implemented several innovative and revolutionary ideas. Till then, a general consensus was the operating systems can only be developed in an assembly language. Though Unix, too, was originally written in assembly language, in 1972 much of it was rewritten in the C language, a high level language developed specifically for it. Though the idea was not entirely new, it was Unix that popularized this novel concept. Assembly languages are inherently tied to particular machine architecture and it is extremely difficult to port (convert) programs written in the assembly language of one computer into the assembly language of another computer having a very different architecture. On the other hand, high level languages are not tied to any specific hardware architecture and all programs written in a high level language become immediately available on a new computer platform as soon as a compiler for the language is developed for the new platform. In those days when there were as many different architectures and operating systems as there were computer manufacturers, this easy portability was a major boon. Earlier, people and organizations had to learn a new operating system every time they purchased a new type of computer. Now they had the option of porting their existing operating system to the new computer architecture and continue working in a familiar environment.

Also, as a part of a court settlement with the US Government, AT&T was not allowed to sell computer software. So they did not object to the Unix developers giving out copies of the Unix operating system with source code and online manuals to others for free. Soon many universities, government agencies and private companies started using Unix. Because, the source code was available, it was easy to make the small amount of changes needed to run Unix on a new platform. It also allowed the universities and organizations to study the source code and enhance it with new features. Unix became phenomenally successful in the subsequent years. Denis Ritchie and Ken Thompson were awarded the Turing Award, considered to be the Nobel Prize of computing, in 1983.

The success of Unix can be largely attributed to the revolutionary concepts it pioneered or popularized for the first time. Some of these are listed below.

- Unix was the first successful operating system to have been developed in a high level language
- Unix popularized the multi-level hierarchical file system. With some modifications, it is still in use by all major operating systems
- Unix simplified device access by abstracting I/O to devices also as file I/O
- Unix provided a very powerful command line environment that supported combining the power of existing commands in flexible ways to get a new job done (I/O redirection). This major innovation dramatically changed the way people worked, improved their efficiency and continues to be a major strength of the platform
- Unix stored all configuration information in plain text files, making them easily accessible and modifiable
- Unix started a new trend by providing an online manual with the system itself, so there was no need to walk to the library to fetch a printed manual if one forgot some command or option.
- The communication and networking technologies that are so ubiquitous today, were largely developed first on Unix systems
- Unix provides powerful pattern matching capabilities in many of the tools that are part of the environment

Advantages of Unix

- Multi-user
- Multitasking
- The Unix Philosophy (do one thing well)
- Hierarchical file system
- Emphasis on the use of plain text files for configuration and data storage
- Devices as files
- Pattern matching
- Shell programming language
- Portability
- Online documentation
- Communication
- Security
- Reliability
- Performance
- Scalability
- Abstracted I/O & I/O redirection

from the beginning, Unix has been a system designed by programmers, for the programmers. Unlike other *user-friendly* operating systems that try to protect the user from committing mistakes unintentionally, Unix generally assumes you know what you are doing. It carries out the command issued by the user without bothering him/her with prompts to confirm the action. Unix commands work silently; usually there is no extraneous output when a commands does its job successfully. Thus applies especially to the command line environment. On the other hand, the GUI interfaces are quite user-friendly.

Unix Turns Commercial

As Unix became extremely popular and AT&T, after its breakup, was allowed to sell computer software, the company trademarked the name UNIX and started charging for the Unix operating system. Many computer companies also purchased Unix from AT&T with source code, modified it and developed their own commercial Unix-like operating systems. This generated a backlash among the community that had received Unix for free so far. Some, especially universities, asserted their right to a free Unix system. The University of California at Berkley developed their own free Unix version called BSD (Berkley Software Distribution) Unix. BSD Unix made several important contributions to Unix. AT&T challenged them in court. After a long legal battle, both commercial and free versions of Unix were permitted.

MINIX

The famous academician and author Andrew S Tanenbaum had developed a Unix-like operating system called MINIX for teaching principles of operating systems. Its source code was published as a part of his textbook Operating Systems: Design and Implementation. However, the publisher had restricted its use to those who purchase the book and to academic use only. Thus, it was not free.

Andrew S Tanenbaum

The Genesis of the Free Software Movement

Around early 1980s, Richard Stallman, working at MIT, got increasingly frustrated by the various restrictions placed by commercial software vendors on use and sharing of computer software. His vision for software was to provide all kinds of freedom to the users. His interpretation of the word free was not free as in free buttermilk i.e. gratis (without payment of money), but it was free as in freedom. He outlined four fundamental types of freedoms for users of software.

- **Freedom 0 :** The freedom to run the program, for any purpose
- **Freedom 1 :** The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.
- **Freedom 2 :** The freedom to redistribute copies so you can help your neighbor
- **Freedom 3 :** The freedom to distribute copies of your modified versions to others

Richard Stallman at an Inauguration in Kolkata

To this end, he started the GNU project in 1983. He wanted GNU to be a Unix-like (in its working), but completely free operating system. To emphasize that GNU was not a commercial system like Unix, he chose the name GNU that stood for GNU is Not Unix. This is called a recursive acronym. Such recursive acronyms are common in the names of computer programs, especially open source ones. By free, he meant all the freedoms mentioned above. He started a Free Software Foundation (FSF) for developing GNU and other entirely free software projects. By 1990, he and the volunteers of the FSF had created most of the major components of the proposed GNU operating system, including the compiler, the shell, the libraries and the utilities. But the core component, the kernel, was unfinished. Work on the kernel was going on, but was slow.

Linux is Born

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu)… – Linus Torvalds (1991)

In 1991, Linus Torvalds, a student from Finland, developed an experimental operating system kernel for the PC (personal computer) called the Linux kernel (Linux stood for Linux is Not Unix, again a reference to Unix's now commercial status). After he opened communications with other programmers on the Internet, the project grew rapidly. With the help of these volunteers, finally the Linux kernel and the GNU components were combined to form the first completely free working Unix-like system (BSD's case was still going on in the courts). The combined system came to be known as GNU/Linux, or simply Linux and became widely popular after Linus granted all the freedoms to everyone (GNU had already done so) and a large community of volunteers, individuals, organizations and even commercial corporations, started supporting its development.

Linus Torvalds

It's All About Freedom and Choice

Linux is a mass movement today. A large community spearheaded by Linus Torvalds himself looks after the development of the Linux kernel. Several other individuals and communities continually work towards providing better software solutions for all common requirements of users with liberal freedoms. In most cases, the freedom includes the rights to obtain the source code of the software, modifying it and contributing the changes back to the community or redistributing the modified software under a name of one's own choosing. As a result, there is a bewildering choice of software available for all the common requirements of computer users. Different people and organizations select bundles of software from this vast pool of software as per their own criteria and preferences and create a distribution of the Linux operating system that they distribute under their own name. Each distribution of Linux is a bundle of some version of the Linux kernel and a set of software applications selected with some goals in mind. There are thousands of Linux distributions, called distros, available now. Some of the more popular ones include Ubuntu Linux, Linux Mint, Fedora Linux, Debian GNU/Linux, Red Hat Enterprise Linux (this is a commercial, but open source, distribution), openSUSE Linux, Knoppix, etc. These distribution vary in their goals as well as contents. The range include from the 12 MB Tiny Core Linux and 100 MB Puppy Linux to the large ones reaching gigabytes in size and bundling practically everything you may need, and then some more. All of them are variants of the same operating system, because all of them

run the Linux kernel. However, due to differences in the selection of the window manager and shell, the looks may differ markedly. Some other operating systems also have some of their major components based on Linux or Unix. For example, both the OS X operating system for personal computers and the iOS operating system for smartphones and tablet computers from Apple Inc. use a BSD Unix subsystem. The Android smartphone operating system from Google uses a slightly modified Linux kernel as well as several Linux libraries.

Free v/s Commercial Software

By now the reader might be wondering why even commercial organizations get involved in the development of software products that are avowedly free. They not only encourage their development from the ringside, they even commit their resources (programmer time as well as money) to such products. Some companies like Red Hat Inc. and Canonical Ltd. exist primarily for developing such products. To understand this, one first needs to study the cost and revenue dynamics of the software world. Let us have a look at both the commercial software model and free software model.

Commercial Software Model

Developing computer software costs; and it costs a lot. In most projects of automation the software cost is substantially higher than the hardware cost. Much of this cost is due to the salaries paid to skilled Information Technology (IT) professionals. Software development is a complex process and risk of failure is also quite high. Under the circumstances, commercial organizations developing software expect substantial returns on their investment. However, once a single copy of the software goes outside the organization, it is ridiculously easy to make any number of copies of the software. This is where the software business is completely different from the brick-and-mortar businesses. In product manufacture, even if you get access to the blueprint for making a product, it is not easy to produce a genuine looking duplicate product because one would also need matching material and machines. On the other hand, while the process of creating the first copy of software is very costly, the process of creating additional copies from it cost next to nothing. Hence software companies try to prevent their customers from making copies of the software and start selling the software themselves.

Usually, the software companies do not provide the original source code developed by their programmers in a high-level language at great efforts to their customers. They only provide the binary machine code or executable version of the programs to the customers. While this is good enough to run on a computer, it is extremely difficult to understand and modify such binary versions of software. If someone takes a

company's software and sells it without any changes as one's own development, the bluff will get caught. If the original developer has asserted one's copyright on one's original work of creation, the copyright laws prevent others from making copies of the work without permission from the copyright holder.

Further, the companies never "sell" the software, they merely license it. Even when you "purchase" legal software, actually you do not purchase the software; you merely purchase a license to use the software subject to a host of terms and conditions. That is why, when you install some software, you are made to click on an "I agree" or "Accept" button. Once you click on this button, you are legally bound to follow the licensing terms that, among other things, prohibit you from making copies and giving to others. A host of other measures like registration keys and Internet based activation are used to prevent piracy (illegal copy and use) of software. Despite all these efforts, piracy does happen on large scale, especially in certain parts of the world and companies have to recover the lost revenue from the genuine purchasers of software.

Free Software Model

There is another side of the coin also. There is a large number of highly skilled persons having an altruistic nature who are ready to spend some part of their time creating free software or helping others out in developing software, without expecting anything in return. Large communities of such "Good Samaritans" collaborating over the Internet can develop (and indeed have developed) even very complex software that would have been quite costly otherwise. Since these people also provide the source code with the software, others get a chance to easily modify the same, correct errors and add new features. Over a period of time such software may not just rival, but even surpass the equivalent commercial offerings. In this environment, there are no restrictions on anybody and everyone enjoys full freedom regarding what one wants to do with the software. This is the vision of Richard Stallman and other advocates of free software. According to Stallman free software does not mean just that you don't have to pay for it. Free, in his version, stands for freedom – you are free to do whatever you want to do with the software; you may use it, you may give it to others legally, you may study its source code, you may modify it and redistribute it under your own name; you have almost all the freedom. However, as a moral duty you should give credit to the original developers and hence you must preserve their copyright notices, while adding your own (even free software contains copyright notices, if only to ascertain the authors and to prevent others from claiming a right on similar work). Also, if you modify the software, you may persuade the original developer to incorporate your modifications in the original software or you must distribute the modified software under a different name; because if any errors were introduced by your modifications, you should be responsible for the same, not the original developer.

Richard Stallman has also advocated another condition on such free software – if you develop some modified or new software based on such free community-developed software, you must also give the software you developed in this way (with source code) back to the community for free. This, according to him, is the only way to sustain the free software movement and continue further development of community software. Software given for free under this condition is called "copyleft" (the exact opposite of copyright). He has designed a software license called GNU GPL (General Public License) that enforces all these conditions and he exhorts everyone to release their software under this license. The license means that if you take some code under the GPL and modify/enhance it, you must distribute your enhanced/modified code also under the same license i.e. the GPL. In other words, you must also provide your modified software along with its source code freely. Hence a commercial entity cannot take *GPLed* software, enhance it and sell the enhanced version commercially to make money. There are groups that do not like such strict "copyleft" restriction. They distribute their software under more "permissive" licenses like the BSD or Apache license that do allow commercial entities to sell software made or enhanced from free software. In a multi-licensing scheme, the software is distributed under more than one license and the user is free to select which license they want to obey.

Different Types of Free Software

Free software come in many different flavors – GPL provides the highest freedom to the users, and also, indirectly, prohibits commercial use. The GPL is considered to be the purest form of free software. Software whose source code is freely available is called open source software. *GPLed* software must also provide the original source code of the software and hence is also open source. Sometimes the binary machine-executable version of the software is provided for free, but the source code is not provided; so modifications cannot be made. Such software is free, but not open source. Sometimes the software is given for a free trial for a certain period or with certain limitations or certain functionalities disabled. If you like the software in the trial, you are supposed to purchase it. The software may stop working after the trial period. This is known as shareware or trialware. Sometimes, even if you do not pay, the software may continue to work fully, but with annoying reminders about the pending payment with a waiting period of few seconds or a minute or two. Such software is sometimes known as nagware.

Sometimes the full software is made available for free, but under some legal restriction such as "for personal/academic/non-commercial use only" and for commercial use one must purchase the software. Some software is free and development is supported by voluntary donations. Some software developers earn from the revenue of advertisements shown in the software or on the maker's web site. Such software is known as adware. Some software is made available for free, but you

are requested to pay or donate a small amount for its use and to support further development. However, if you do not pay, the software continues to work fully.

The Free Software Revenue Model

While companies selling proprietary software earn from the sales revenue, companies working with free software of various categories have different revenue models. Some of them earn money by providing “training and support services” on a commercial basis. Due to the complexity, large software often has several bugs or errors in them. An individual encountering such errors may not suffer much materially. The person may get help from friends, from Internet forums and try to solve or bypass the error on one’s own. This may sometimes take a long time also. But for commercial organizations if the computer systems don’t work for some time, it may incur heavy financial losses. Hence they prefer the assurance of getting training and support from qualified engineers, often the very people who have developed or contributed significantly to the software. Such organizations are even ready to pay for such support, if it turns out to be cheaper than buying proprietary software. Hence giving the software for free to everyone and charging corporate entities for support is one strategy. Canonical Ltd. has followed this strategy for Ubuntu.

Other strategies include displaying online advertisements and earning revenue from them, giving a basic version free and charging for a “premium” version with more features, etc. The latter model is named "freemium". Companies can even make money by collecting contact details (like email id) through a “free registration” process and then selling these contact details to mass advertisers. Some companies support a completely free and open source version of particular software, but derive indirect benefits by incorporating the enhancements made to the free version by the community into their own commercial offerings. For example, Red Hat Inc., another major Linux company, started and supports the free and open source Fedora Linux distribution in a major way. Enhancements made in Fedora Linux by the collaborative work of Red Hat engineers and thousands of volunteers around the world often find their way into the commercial Red Hat Enterprise Linux product.

Do you still think the commercial interests of a for-profit company and free products are just not compatible with each other? Consider these facts –

- On 19th January in the year 2000, shares in Yahoo! Japan became the first stock in Japanese history to trade at over ¥100,000,000, reaching a price of 101.4 million yen ($9,62,140 at that time) for a single share. All of the company’s products and services were free.
- In the year 2016, Google’s revenues from advertising stood at $79 billion.

- On 18th May 2012, when Facebook entered the stock market for the first time, the company was valued at $104 billion. The company provides free social networking services to its users.

Proprietary software v/s open source software

- Cost
- Freedom
- Bugs, patches, vulnerabilities
- Features
- Compatibility
- Choice

Why commercial organizations take part in free software development

- Earning by providing support
  - Total Cost of Ownership (TCO) Cost of purchase, maintenance, training
- Advertisements
- Profiling users and collecting personal information
  - Targeted advertisements
  - Machine learning
- Parallel versions (e.g. RedHat Enterprise Linux and Fedora Linux)
  - Community edition v/s enterprise edition
- Benefiting from side effects (e.g. Android)
- Developing software for internal use and then releasing as open source

Different categories of free software

- Free and open source software (FOSS)
- Free but not open source
- Trial versions (limitation - time, features)
- Full, but legally limited versions
- only for non-commercial use
- Shareware
- Adware
- Nagware

Different types of licenses and IPR (Intellectual Property Rights) issues

- IPR
  - Patent
  - Copyright

    - Trademark (logos, names, distinctive style)
- Proprietary licenses
- Free licenses
    - Copyleft / restrictive licenses (GPL)
    - Permissive / liberal licenses (Apache, BSD, MIT, Mozilla, etc.)
    - Creative Commons licenses
- Other intellectual assets
    - text
    - images
    - audio
    - video
    - animation

Comparison of Popular Operating Systems for Personal Computers

When we compare popular operating systems for the personal computers, several differences prop up. There are three major families of operating systems for the PC. The proprietary Microsoft Windows family of operating systems is by far the most popular. Apple Computer's Mac series computers use their own proprietary operating system called OS X (earlier it was the MAC OS). These computers have found a niche in the premium brand-conscious segment where they have a fierce fan following. The third operating system is Linux. Linux is used by a variety of people, from the geeks (very technically minded people) to people who want to support the free software movement to people owning older or lower configuration hardware that won't run the other operating systems properly or people who can't or don't want to pay for an operating system. Each OS has its own pros and cons.

Microsoft Windows

Microsoft Windows is a proprietary OS and must be purchased for using it. It is a very user-friendly, easy to use OS and almost everyone is familiar with it. Because of its popularity, a large number of commercial as well as free software is available for the OS. It also boasts of an excellent device driver support. Even though Microsoft has taken significant strides in improving its stability and security compared to early versions, lingering doubts about these issues still remain. Being the most popular OS among users has also made it the most popular target of attacks by crackers (highly skilled programmers with malicious intention). Malware like computer viruses, worms, Trojans and several others continue to affect systems running Microsoft Windows. Even using a top-notch commercial security solution does not seem to be able make one's system completely immune to such attacks. Security of Microsoft

Windows systems is a continuous headache for large organizations. Microsoft Windows is also infamous for needing too much hardware resources and higher-end configurations to run decently. As people run a mix of software from so many different sources on Microsoft Windows systems, the overall experience is somewhat varied and when there is a problem, it becomes difficult to pinpoint the exact source of the trouble. Microsoft Windows officially supports only the built-in shells for the GUI (Windows Explorer) and CLI, though third party alternative are available.

Apple OS X

Apple OS X is also a proprietary operating system. It comes bundled with the machine and neither work without the other (barring some third party experimental solutions). The system is known for its high quality hardware and visual appeal and what ardent Apple fans believe to be the best user experience. The entire hardware, operating system and software environment is tightly controlled by Apple to provide a highly consistent and reliable user experience. OS X also officially supports only the built-in GUI and CLI, though third party alternatives are available to try. Though the systems are considered quite secure; it, too, is not immune from attacks. The major advantage of the system is its consistent, high-quality user experience. Against that, the user is confined to a narrow world controlled by Apple and third party application support is limited. Also, the product comes at a high premium. Internally, OS X is a graphical environment that runs on top of a Mach kernel and a BSD Unix subsystem.

Linux

Linux, just like its predecessor Unix, is known for its high performance, security, reliability and portability. No matter how antiquated or low-end the hardware is, one can find a Linux distribution that would run on it. Even though lack of whole-hearted support by BIOS and device vendors in providing high quality device drivers dents Linux's image of being a reliable and portable OS to some extent, it is quite stable on most PC configurations. It combines the high-power CLI that has traditionally been Unix's strength with an impressive GUI that makes it almost as user-friendly as the other two OSs. Linux lets us choose from a wide variety of distros, then presents an even wider array of mostly free applications to install from and is flexible enough that with experience we can configure it exactly the way we want it to be. However, free applications available under Linux sometimes lack the polish of corresponding Windows applications (especially on the GUI front) and often may not provide all the bells and whistles that commercial applications on Microsoft Windows or OS-X platforms provide. But they do get the job done most of the time and keep getting improvements from the community. Upgrades are as free as the base OS and it is just a matter of user's convenience when they want to upgrade their systems. However, one should be prepared for the occasional glitches. Linux provide a number of choices

for the CLI as well as GUI. The most common GUIs are KDE, GNOME and Unity. The newest versions of the GUIs provide a strong competition to the other two OSs as far as visual attractiveness is concerned, though they, too, require a bit higher configuration.

A major advantage of Linux is that not only is it free; it also provides us complete freedom in running the operating system. Unlike other operating systems that require at least one free hard disk partition (that would be erased during the installation) and a tedious installation process, there is a range of choices for running Linux – from a no-risk no-installation trial using a Live CD to installing Linux alongside Windows (multi-boot configuration) to booting Linux directly off a USB flash disk or even the network – Linux supports it all. Linux also provides excellent interoperability with Microsoft Windows and you can access the Windows partitions on your computer and the organization's Microsoft Windows servers as easily as from Windows. Unfortunately, facility of accessing the Linux partition from Microsoft Windows is generally limited to read-only access (we can read the files but cannot modify them) using third party tools. The office suite on Linux – OpenOffice.org or LibreOffice provides good interoperability with the Microsoft Office suite with few caveats and some irritants. In essence, Linux meets all the basic needs of an average computer user and, once you settle down, can be as comfortable to use as it can get. And all this is yours, 100% free, 100% legal, no strings attached.

The market share of Linux is difficult to estimate because it is free and there is no central point of distribution, but Linux has a very dominating presence in supercomputers, mainframes, servers and graphics workstation markets. While much of the PC market is dominated by Microsoft Windows and OS X has a considerable presence in the premium segment and has its own loyal fan base, Linux is strong in some organizational setups and at the lower end of the spectrum, with low-end PCs. It is also popular among developers. Linux, too, has developed its own fan base who sneer at people using any other operating system.

Introduction to Ubuntu Linux

A traveler through a country would stop at a village and he didn't have to ask for food or for water. Once he stops, the people give him food, entertain him. That is one aspect of Ubuntu, but it will have various aspects. Ubuntu does not mean that people should not enrich themselves. The question therefore is: Are you going to do so in order to enable the community around you to be able to improve? – Nelson Mandela (anti-apartheid crusader, freedom fighter and former president of South Africa)

Ubuntu Linux is a Linux distribution created by the UK based company Canonical Ltd., established by the South African entrepreneur Mark Shuttleworth. It is in turn based on the Debian GNU/Linux distribution. Ubuntu is an ancient African word meaning 'humanity to others'. It is a philosophy that emphasizes putting common goals and the community above individual interests and believes in helping one another. As the open source software community also has similar philosophy, Ubuntu was chosen as the name of the distribution (if you want to know one more reason for choosing this name, read the two words after the second “the” in the first line of this paragraph again). Ubuntu is free and open source software. Canonical expects to earn money by providing paid support services. While Canonical is the main sponsor, Ubuntu is also supported by the Ubuntu Foundation and large developer and user communities. Ubuntu focuses on usability, security and stability. Its focus on usability (ease of use) and good device support has allowed it to gain and retain a place among the top Linux distributions.

Ubuntu Versions

Ubuntu has a fixed release cycle, with a new version being released in the April and October months of each year. The release numbers are denoted by two-digit year, followed by a dot, followed by two-digit month. Thus, Ubuntu 12.04 LTS was released in April 2012. The releases also have two-word names, with the first word being an adjective and the second the name of an animal. For example, Ubuntu 12.04 LTS was called Precise Pangolin, Ubuntu 12.10 was called Quantal Quetzal. Ubuntu 16.10 is called Yakkety Yak. People often use only the first word to identify the release. The first words are chosen in alphabetic order, so one can know which version is newer just by looking at the first letter of the name. Each desktop edition release is officially supported for 9 months. Every two years, a Long Term Support (LTS) version is released. These versions are supported for 5 years.

Version	Code name	Release date	Supported until
14.04 LTS	Trusty Tahr	2014-04-17	2019-04
14.10	Utopic Unicorn	2014-10-23	2015-07
15.04	Vivid Vervet	2015-04-23	2016-02
15.10	Wily Werewolf	2015-10-22	2016-07
16.04 LTS	Xenial Xerus	2016-04-21	2021-04
16.10	Yakkety Yak	2016-10-13	2017-07
17.04	Zesty Zapus	2017-04-13	2018-01
17.10	Artful Aardvark	2017-10-19	2018-07
18.04 LTS	Bionic Beaver	2018-04-26*	2023-04

There are several other variants of Ubuntu including a server edition, Edubuntu (a distro aimed at school students, with built-in educational software), Kubuntu / Ubuntu GNOME (variants using the KDE / GNOME desktop instead of the default Unity desktop), Lubuntu / Xubuntu (lightweight variants for use on low-end computers using the LXDE / Xfce desktop environment), etc.