Linux: the big picture

Abstract

This article gives a brief introduction to Linux, with a sketch of the background history. It was written for *PC Update*, the monthly magazine of the PC user group in Melbourne, Australia. This version does not contain all the editorial changes and spelling fixes made by the magazine.

April 28, 2003, Lars Wirzenius (liw@iki.fi).

Text taken from http://liw.iki.fi/liw/texts/linux-the-big-picture.html, August 7, 2006

History

The history of computer operating systems starts in the 1950s, with simple schemes for running batch programs efficiently, minimizing idle time between programs. A batch program is one that does not interact with the user at all. It reads all its input from a file (possibly a stack of punch cards) and outputs all its output to another file (possibly to a printer). This is how all computers used to work.

Then, in early 1960s, interactive use started to gain ground. Not only interactive use, but having several people use the same computer at the same time, from different terminals. Such systems were called time-sharing systems and were quite a challenge to implement compared to the batch systems.

During the 1960s there were many attempts at building good time-sharing systems. Some of these were university research projects, others were commercial ones. One such project was Multics, which was quite innovative at the time. It had, for example, a hierarchical file system, something taken for granted in modern operating systems.

The Multics project did not, however, progress very well. It took years longer to complete than anticipated and never got a significant share of the operating system market. One of the participants, Bell Labs, withdrew from the project. The Bell Labs people who were involved then made their own operating system and called it Unix.

Unix was originally distributed for free and gained much popularity in universities. Later, it got an implementation of the TCP/IP protocol stack and was adopted as the operating system of choice for early workstations.

By 1990, Unix had a strong position in the server market and was especially strong in universities. Most universities had Unix systems and computer science students were exposed to them. Many of them wanted to run Unix on their own computers as well. Unfortunately, by that time, Unix had become commercial and rather expensive. About the only cheap option was Minix, a limited Unix-like system written by Andrew

Tanenbaum for teaching purposes. There was also 386BSD, a precursor NetBSD, FreeBSD, and OpenBSD, but that wasn't mature yet, and required higher end hardware than many had at home.

Into this scene came Linux, in October, 1991. Linus Torvalds, the author, had used Unix at the University of Helsinki, and wanted something similar on his PC at home. Since the commercial alternatives were way too expensive, he started out with Minix, but wanted something better and soon started to write his own operating system. After its first release, it soon attracted the attention of several other hackers. While Linux initially was not really useful except as a toy, it soon gathered enough features to be interesting even for people uninterested in operating system development.

Linux itself is only the kernel of an operating system. The kernel is the part that makes all other programs run. It implements multitasking, and manages hardware devices, and generally enables applications to do their thing. All the programs that the user (or system administrator) actually interacts with are run on top of the kernel. Some of these are essential: for example, a command line interpreter (or shell), which is used both interactively and to write shell scripts (corresponding to .BAT files).

Linus did not write these programs himself, and used existing free versions instead. This reduced greatly the amount of work he had to do to get a working environment. In fact, he often changed the kernel to make it easier to get the existing programs to run on Linux, instead of the other way around.

Most of the critically important system software, including the C compiler, came from the Free Software Foundation's GNU project. Started in 1984, the GNU project aims to develop an entire Unix-like operating system that is completely free. To credit them, many people like to refer to a Linux system as a GNU/Linux system. (GNU has their own kernel as well.)

During 1992 and 1993, the Linux kernel gathered all the necessary features it required to work as a replacement for Unix workstations, including TCP/IP networking and a graphical windowing system (the X Window System). Linux also received plenty of industry attention, and several small companies were started to develop and distribute Linux. Dozens of user groups were founded, and the Linux Journal magazine started to appear in early 1994.

Version 1.0 of the Linux kernel was released in March, 1994. Since then, the kernel has gone through many development cycles, each culminating in a stable version. Each development cycle has taken a year or three, and has involved redesigning and rewriting large parts of the kernel to deal with changes in hardware (for example, new ways to connect peripherals, such as USB) and to meet increased speed requirements as people apply Linux to larger and larger systems (or smaller and smaller ones: embedded Linux is becoming a hot topic).

From a marketing and political point of view, after the 1.0 release the next huge step happened in 1997, when Netscape decided to release their web browser as free software (the term 'open source' was created for this). This was the occasion that first brought free software to the attention of the whole computing world for the time. It has taken years of work since then, but free software (whether called that or open source) has become not only generally accepted but also often the preferred choice for many applications.

Social phenomenon

Apart from being a technological feat, Linux is also an interesting social phenomenon. Much through Linux, the free software movement has broken through to general attention. On the way, it even got an informal marketing department and brand: open source. It is baffling to many outsiders that something as successful as Linux could be developed by a bunch of unorganized people in their free time.

The major factor here is the availability of all the source code to the system, plus a copyright license that allows modifications to be made and distributed. When the system has many programmers among its users, if they find a problem, they can fairly easily fix it. Additionally, if they think a feature is missing, they can add it themselves. For some reason, that is something programmers like to do, even if they're not paid for it: they have an itch (a need), so they scratch (write the code to fill the need).

It is necessary to have at least one committed developer who puts in lots of effort. After a while, however, once there are enough programmer-users sending small changes and improvements, you get a snowball effect: lots of small changes result in a fairly rapid total development speed, which then attracts more users, some of which will be programmers. This then results in more small changes and improvements sent in by users, and so on.

For operating system development specifically, this large group of programmer-users results in two important types of improvements: bug fixes and device drivers. Operating system code often has bugs that only occur rarely and it can be difficult for the developers to reproduce them. When there are thousands or more users who are also programmers, this results in a very effective testing and debugging army.

Most of the code volume in Linux is device drivers. The core functionality, which implements multitasking and multiuser functionality, is small in comparison. Most device drivers are independent from each other, and only interact with the operating system core via well defined interfaces. Thus, it is fairly easy to write a new device driver without having to understand the whole complexity of the operating system. This also allows the main developers to concentrate on the core functiionality, and they can let those people write the device drivers who actually have the devices.

It would be awkward just to store the thousands of different sound cards, Ethernet cards, IDE controllers, motherboards, digital cameras, printers, and so on that Linux supports.

The Linux development model is distributed, and spreads the work around quite effectively.

The Linux model is not without problems. When a new device gets on the market, it can take a few months before a Linux programmer is interested enough to write a device driver. Also, some device manufacturers, for whatever reason, do not want to release programming information for their devices, which can prevent a Linux device driver to be written at all. Luckily, with the growing global interest in Linux such companies become fewer in numbers.

What it is

Linux is a Unix-like multitasking, multiuser 32 and 64 bit operating system for a variety of hardware platforms and licensed under an open source license. This is a somewhat accurate but rather brief description. I'll spend the rest of this article expounding on it.

Being Unix-like means emulating the Unix operating system interfaces so that programs written for Unix will work for Linux merely by re-compiling. It follows that Linux uses mostly the same abstractions as the Unix system. For example, the way processes are created and controlled is the same in Unix and Linux.

There are a number of other operating systems in active use: from Microsoft's family of Windows versions, through Apple's MacOS to OpenVMS. Linux's creator, Linus Torvalds, chose Unix as the model for Linux partly for its aesthetic appeal to system programmers, partly because of all the operating systems he was familiar with, it was the one he knew best.

The Unix heritage also gives Linux the two most important features: multitasking and multiuser capabilities. Linux, like Unix, was designed from the start to run multiple processes independently of each other. Implementing multitasking well requires attention at every level of the operating system. It is hard to add multitasking to an operationg system afterwards. That's why the Windows 95 series and MacOS (before MacOS X) did multitasking somewhat poorly: multitasking was added to an existing operating system, not designed into a new one. That's also why the Windows NT series, MacOS X, and Linux do multitasking so much better.

A good implementation of multitasking requires, among other things, proper memory management. The operating system must use the memory protection support in the processor to protect running programs from each other. Otherwise a buggy program (that is, most any program) may corrupt the memory area of another program, or the operating system itself, causing weird behavior or a total system crash, with likely loss of data and unsaved work.

Supporting many concurrent users is easy after multitasking works. You label each instance of a running program with a particular user and prevent the program from tampering with other user's files.

Portable and scalable

Linux was originally written for an Intel 386 processor, and naturally works on all successive processors. After about three years of development, work began to adapt (or port) Linux to other processor families as well. The first one was the Alpha processor, then developed and sold by the Digital Equipment Corporation. The Alpha was chosen because Digital graciously donated a system to Linus. Soon other porting efforts followed. Today, Linux also runs on Sun SPARC and UltraSPARC, Motorola 68000, PowerPC, PowerPC64, ARM, Hitachi SuperH, IBM S/390, MIPS, HP PA-RISC, Intel IA-64, DEC VAX, AMD x86-64 and CRIS processors. (See kernel.org for details.)

Most of those processors are not very common on people's desks. For example, S/390 is IBM's big mainframe architecture. Here, mainframe means the kind of computer inside of which you can put your desk, rather than the kind that fits on your desk.

Some of those processors are 32 bit, like the Intel 386. Others are 64 bit, such as the Alpha. Supporting such different processors has been good for Linux. It has required designing the system to use proper modularity and good abstractions and this has improved code quality.

The large variety of supported processors also shows off Linux's scalability: it works everything from very small systems, such as embedded computers, handheld devices, and mobile phones, to very large systems, such as the IBM mainframes.

Using clustering technology, such as Beowulf (beowulf.org), Linux even runs on supercomputers. For example, the US Lawrence Livermore National Laboratories bought a cluster with 1920 processors, resulting in one of the five fastest supercomputers in the world with a theoretical peak performance of 9.2 teraFLOPS or 9.2 trillion calculations per second. (LWN article).

Using Linux

The operating system itself is pretty boring to most people. Applications are necessary so to get things done. Traditionally, Linux applications have been the kinds of applications used with Unix: scientific software, databases, and network services. Also, of course, all the tools programmers want for their craft.

Much of such software seems rather old-fashioned by today's desktop standards. User interfaces are text based, or they might not exist at all. Indeed, most software has usually been non-interactive and has been of the command line, batch processing variety. Since most users have been experts in the application domain, this has been good enough.

Thus, Linux first found corporate employment as a file server, mail server, web server, or firewall. It was a good platform for running a database, with support from all major commercial database manufacturers.

In the past few years Linux has also become an interesting option on the user friendly desktop front. The KDE (kde.org) and Gnome (gnome.org) projects develop desktop environments and applications that are easy to learn (as well as effective to use). There is now plenty of desktop applications which people with Windows or MacOS experience will have no difficulty using.

There is even a professional grade office software package. OpenOffice (openoffice.org), based on Sun's StarOffice, is free, fully featured, and file compatible with Microsoft Office. It includes a word processor, spreadsheet, and presentation program, competing with Microsoft's Word, Excel, and Powerpoint.

Linux distributions

To install Linux, you have to choose a Linux distribution. A distribution is the Linux kernel, plus an installation program, plus some set of applications to run on top of it. There are hundreds of Linux distributions, serving different needs.

All distributions use pretty much the same actual software, but they are different in which software they include, which versions they pick (a stable version known to work well or the latest version with all the bells and whistles and bugs), how the software is pre-configured, and how the system is installed and managed. For example, OpenOffice, Mozilla (web browser), KDE and Gnome (desktop environments), and Apache (web server) will all work on all distributions.

Some distributions aim to be general purpose, but most of them are task specific: they are meant for running a firewall, a web kiosk, or meant for users within a particular university or country. Those looking for their first Linux experience can concentrate on the three biggest general purpose distributions: Red Hat, SuSE, and Debian.

The Red Hat and SuSE distributions are produced by companies by the same names. They aim at providing an easy installation procedure, and for a pleasant desktop experience. They are also good as servers. Both are sold in boxes, with an installation CD and printed manual. Both can also be downloaded via the network.

The Debian distribution is produced by a volunteer organization. It's installation is less easy: you have to answer questions during the installation the other distributions deduce automatically. Nothing complicated as such, but requiring understanding of and information about hardware most PC users don't want to worry about. On the other hand, after installation, Debian can be upgraded to each new release without re-installing anything.

The easiest way to try out Linux is to use a distribution that works completely off a CD-ROM. This way, you don't have to install anything. You merely download the CD-ROM image from the net and burn it on a disk, or buy a mass-produced one via the net. Insert disk in drive, then reboot. Not having to install anything on the hard disk means you can easily switch between Linux and Windows. Also, since all Linux files are on a read-only CD-ROM, you can't break anthing by mistake while you're learning.

Further information

- The Linux Documentation Project. A good place to start looking for manuals and help.
- Linux Weekly News, Slashdot: Two important news sites about Linux and related things.
- Linux distributions: Debian, Red Hat, SuSE.
- Knoppix: Single bootable CD, for testing Linux without installing.

Author blurb

Lars Wirzenius designs and implements embedded telematic software for Oliotalo at work, and develops Debian at home.