Roots of a revo

Linux has made a remarkable impression on industry over the past few years. Linux history, however, goes back almost a decade, or more than 17 years if you count the efforts of the Free Software Foundation. Linux is built on the paradigms, although not the source code, of Unix systems. Today Linux is still perceived by some as a new and curious beast rather than a proven and solid Unix like solution. In fact, it is not Linux that is new, but the popularity of Linux.

One probably has to go back all the way to 1969 in order to understand the importance of Linux to the computer industry. In an attempt to overcome the restrictions of the then popular batch systems, Ken Thompson started in 1969 to develop the first Unix system at the Bell Laboratories, a daughter of AT&T and Western Electric. This initial version was written in the DEC PDP-7 assembler language. In order to achieve a more machine-independent architecture for his operating system, Thompson developed the programming language B, which was later refined by Dennis Ritchie to the C programming language. The second version of Unix, this time for the PDP-11, appeared in 1971 and was mostly written in C. Most operating systems today are still written in this language, although usually a somewhat more modern dialect is being used, called ANSI C.

The pioneers: Dennis Ritchie and Ken Thompson at work in 1970 (top), Richard Stallman (AKA RMS) in his guise as St IGNUtius (middle), the patron saint of free software, and Linus Torvalds, whose copyleft kernel provided the crucial last piece in the puzzle

A Unix system is defined, not only by the operating system kernel responsible for the communication with the machine's hardware and tasks like scheduling processes, allocating memory and so on, but also by a series of utility programs. These allow the user to perform tasks such as browse the file system, erase or create new files, or to work with text. Utility programs are usually small and specialised so they can be implemented with as little code as possible. This reduces the amount of errors in the code and is one of the main reasons for the remarkable stability and ease of use of a Unix system. Utility programs can be lined up as a sort of chain, where the output of one program forms the input of another. This way you can perform the most complex tasks, while still using the same easy-to-use utility programs.

Did you know?
Linus Torvalds' original 'official' name for his creation was FREAX! Ari Lemmke, the staff member at Helsinki University who set up an FTP site for the code, disliked the name, and used Linux instead. The name appears to have stuck

Almost all commercial Unix systems today are descendants of Unix I Version 7, which appeared in 1979. As of that time the development of Unix split into various branches, the most important of which are System V (or Unix V) and BSD. The latter was developed at the University of Berkeley, California. More surprisingly, for those who weren't alive at the time, there is even a branch of Unix that was started and maintained by Microsoft called XENIX, although Microsoft no longer has any direct involvement. Today there are many commercial Unix systems, the best known of which include IBM AIX, SunOS, Sun Solaris, Compaq Tru64, HP-UX and SGI Irix. There are also several variants of BSD Unix that are freely available.

Another Unix-like operating system that will play a role later in this story, is Minix, introduced by Andrew S. Tanenbaum in January 1987 and designed as a teaching system. Mr. Tanenbaum is a leading computer scientist who now works at the Vrije Universiteit in Amsterdam.

Standardisation and the OS market

Given this wide variety of different Unix implementations, it is not surprising that there was a need for standardisation. Different vendors had different preferences and were often forced to introduce new, sometimes incompatible, features in order to differentiate themselves from the rest of the market. Three main standardisation efforts were started over time. These were the SVID (System V Interface Definition), POSIX (formerly "/usr/group") and X/Open (formerly BISON). X/Open is the most recent attempt to achieve a source code-level compatibility between the participating Unix implementations. Funnily enough it started in Europe, although US companies (including AT&T) joined later. Unfortunately, despite all of these efforts there are still many remaining differences between the different implementations.

During the late 1980s and early 1990s Intel-based computers and the Windows operating system in its various incarnations grew into the enterprise, gaining the major portion of the desktop market and a substantial share of the server market – the traditional domain of Unix systems. Today Intel and Intel-compatible machines can claim to perform in the same league as dedicated Unix workstations with similar computing power. Still, most commercial Unix systems are not available on Intel hardware. Many Unix manufacturers co-operate closely with or own part of the companies that manufacture the chips

lution

Many people just fit Linux and forget it. For those who'd like to know a little of the history, **Ruediger Berlich** charts some of the significant events and motives that drove the development of the free OS

they use, and thus have a vested interest in promoting their hardware. SGI and the MIPS processor line is an example. There are few commercial Unix implementations for Intel machines.

This is where Linux becomes a significant force for business. Not only is Linux a Unix-compatible operating system that challenges Microsoft Windows on its native Intel platforms, but Linux also runs on all major hardware platforms, from handheld devices with Motorola 68000 processors through Intel(-compatible) PCs all the way to IBM S/390 mainframes. Because the same API (application programming interface) is used on every platform, there is complete source code compatibility from platform to platform. Learn it once and use it everywhere. Linux achieves what SVID, POSIX and X/Open strived to achieve for Unix, a common code base for every platform. For an old Unix adept this must be like a dream come true.

Join the rebellion

You might think that after this longish introduction we have finally come to the point where Linus Torvalds unpacks his new and shiny 386SX and starts writing Linux. You are wrong. Technically, Linux history doesn't begin with Linux, at least if you understand Linux to be more than just the kernel.

Linux history begins with Richard Stallman in September 1983. Stallman was a researcher at the Massachussets Institute for Technology (MIT) in Cambridge, Massachussets, when he made the initial announcement of GNU (see box, Here is the GNUs), and what became the Free Software Foundation. Stallman was already known as the author of a text editor called Emacs. Those with long memories of Unix will remember the holy wars between users of the text editors vi and Emacs.

Writing the components of a free operating system takes time. One of the most important parts of the original GNU system, the operating system kernel, known as the Hurd, only saw the light of day very recently (except as an experimental system). However, Richard Stallman made available a multitude of utilities, without which any kind of operating system is useless. Most notably, apart from the text editor Emacs, Stallman developed a compiler called gcc (GNU C Compiler). As one of the major design targets for this compiler was portability, you will today find versions for any operating system under the sun. There are also GNU compilers for many other programming languages including C++, Pascal and Fortran. The GNU utilities were essential to the development of Linux.

GNU, free software and open source

Arguably Stallman's biggest achievement, however, is the creation of the GNU General Public Licence (GPL). All GNU software, by definition, is covered by this licence or the less restrictive LGPL. In summary the GPL says that whenever you make changes to a piece of software covered by the GPL, these changes must also be made available under the GPL. While you may sell software covered by the GPL, you must make available the complete source code to everybody who wishes to gain access. Furthermore, any person using GPL'd software gains the same right to redistribute and modify this software. As the GPL is designed to retain your freedom to do with the software (almost) what you want, it is often called the 'copyleft'.

Richard Stallman and the GPL have initiated a whole new culture of software development, variously called the free software or open source movement. Any person who is willing and capable of providing changes to a piece of GPL'd software can do so. And indeed, as the past 17 years have shown, the GPL has led to a rapid evolutionary improvement in the quality of GPL'd software.

It is important to note that Linux was not the first software project to use the open source paradigm, although it is arguably the most important such project to date.

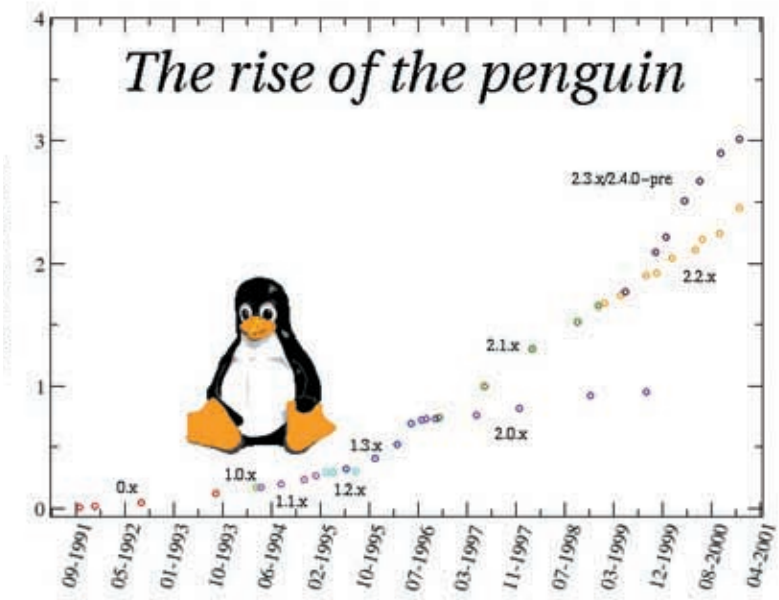
The rise of the penguin

The number of lines of code are shown for the various kernel versions. As you can see, the lines of code in stable kernel versions (2.0, 2.2) increase much slower than the lines of code in the developer kernels (2.1, 2.3). The latest kernel, version 2.4, has more than three million lines of code, compared to 8400 lines of code in the very first version 0.01. Kernel 1.0 (with 170,000 lines of code) was a big achievement

Just a hobby...

At last, we get to the beginning. It's 1991. Trying to gain hands-on experience in operating system design, Linus Torvalds, a student of computer science at the University of Helsinki, Finland, started to work on a new operating system that would later be named Linux after its founder. However, the initial releases were called FREAX, not Linux. This name can still be found in the kernel/Makefile of version 0.11, and other code. The initial development of Linux was done on a relatively low-spec'd 386SX. Linus Torvalds initial posting was to the newsgroup comp.os.minix on 25 August 1991. (See Box, Linus tells the world). At the time of this posting, version 0.01, developed on a relatively low-spec'd 386sx, wasn't actually released and Linux, or FREAX, had very little functionality.

In another posting (26 August 1991) Linus makes a first commitment to the GNU licence: "...Even then it probably won't be able to do much more than minix, and much less in some respects. It will be free though (probably under GNU-license or similar) ...".



Linus tells the world

```
25 Aug 91 20:57:08 GMT
Hello everybody out there using minix -
   I'm doing a (free) operating system (just a hobby, won't be big and profes-
sional 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).a
   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, and I'd like to
know what features most people would want. Any suggestions are welcome, but I won't
promise I'll implement them :-)

          Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT
portable (uses 386 task switching etc), and it probably never will support any-
thing other than AT-hard disks, as that's all I have :-(.
```

Here is the GNUs - RMS announces his free system project

```
Tue, 27-Sep-83 12:35:59 EST

Free Unix!
Starting this Thanksgiving I am going to write a complete Unix-compatible soft-
ware system called GNU (for GNU's Not Unix), and give it away free(1) to everyone
who can use it. Contributions of time, money, programs and equipment are greatly
needed.
   To begin with, GNU will be a kernel plus all the utilities needed to write and
run C programs: editor, shell, C compiler, linker, assembler, and a few other things.
After this we will add a text formatter, a YACC, an Empire game, a spreadsheet,
and hundreds of other things. We hope to supply, eventually, everything useful
that normally comes with a Unix system, and anything else useful, including on-
line and hardcopy documentation.
   GNU will be able to run Unix programs, but will not be identical to Unix. We
will make all improvements that are convenient, based on our experience with other
operating systems.
```

Read the announcement in full at **www.gnu.org/gnu/initial-announcement.html**

The fact that Linus made his code available over the Internet is crucial for the further development of Linux. Many people joined and helped out with feature requests or, better, their own implementation of extensions or features. It is worth noting that use of the Internet was widespread in 1991, so contributions usually came from people that were themselves relatively technical.

It took more than two years for Linux to reach version 1.0, on 16 April 1994. This is not to say that Linux wasn't usable before that time. You will find many who were at university at that time studied and used Linux for all sorts of tasks.

As of version 1.0 the development of the Linux kernel was split. Even version numbers (such as 1.0, 1.2, ...) denote stable kernels ready for use. Odd version numbers (such as 1.1, 1.3, ...) denote developer kernels which are for development purposes only. Once all desired features have been incorporated into a developer series of kernels, the final version undergoes a code freeze, eventually being renamed as the first kernel of a stable series. Usually only bug-fixes are allowed in the stable kernel series, although occasionally important new features get ported back from the developer series.

From the earliest days, the development of Linux depended on many other individuals who devoted their time, skill and considerable effort to the project (see Box, Some key contributors).

Over time, many new features have been integrated into the kernel. The SMP (symmetric multiprocessing) capability was introduced as of kernel 2.0, although it had long been available as an experimental system. Loadable modules were first introduced by Peter McDonald for an 0.99 kernel, although the actual implementation that is in use today is different. As of kernel 1.2 (6.3.1995) Linux availability was formally extended to Alpha, Sparc and MIPS processors, thus overcoming the strong dependency on x86 processors of the earlier versions of the kernel. Today, version 2.4 is available on every major processor platform, from the low-end Motorola 68000 to the IBM S/390 mainframe.

The biggest change from an end user perspective, however, were the new drivers. While initially there was not much more than a driver for a standard IDE drive, later versions introduced support for all kinds of hardware. Crucial to this development was the wide support of Linux by voluntary developers all over the Internet and the willingness of hardware manufacturers to make the specifications for their hardware publicly available. Not all manufacturers have been forthcoming. However, current Linux hardware support need fear no comparison. Kernel 2.4 even adds support for USB and firewire drivers. Nonetheless, continued support from hardware manufacturers remains crucial to the future growth of Linux.

In the next issue, Ruediger concludes his two-part series with a look back at the first Linux distributions, the standards debate, and the appearance of applications. Ruediger Berlich is managing director of SuSE Linux UK

Some key contributors

Werner Almesberger	floppy drivers, LILO
Theodore Ts'o	Ext2 filesystem, libraries, kernel memory allocator, many suggestions to achieve POSIX compatibility
Donald Becker	network drivers (later also Beowulf)
Olaf Kirch	Linux Networking book, NFS code
Alan Cox	networking, a lot of kernel work, early SMP. Alan Cox also maintains the 2.2 kernel series and is one of the main kernel developers today

Of course, all of these people have done much more work than the examples given, and there are countless more developers not mentioned in this short list who have made significant contributions.

Counting from 0.01 to 1.0

Version	Date	Comments
0.01	09/1991	no (binary-)programs available, some device drivers and disk drivers
0.03	26/10/1991	considered usable, shell available, C compiler plus a few utilities
0.12	05/01/1992	the first version distributed under the GPL
0.96	04/1992	the first version capable of running the X Window System
0.99.14	12/1993	the 0.99 series had many sub-versions, as Linux geared up to version 1.0
1.0	16/04/1994	1.0 was released after more than two years of development