



# Linux Kernel Development

## **How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It**

by Greg Kroah-Hartman, SuSE Labs / Novell Inc., [gregkh@novell.com](mailto:gregkh@novell.com)

Jonathan Corbet, LWN.net, [corbet@lwn.net](mailto:corbet@lwn.net)

Amanda McPherson, The Linux Foundation, [amanda@linux-foundation.org](mailto:amanda@linux-foundation.org)

*The kernel which forms the core of the Linux system is the result of one of the largest cooperative software projects ever attempted. Regular 2-3 month releases deliver stable updates to Linux users, each with significant new features, added device support, and improved performance. The rate of change in the kernel is high and increasing, with almost 10,000 patches going into recent kernel releases. These releases each contain the work of nearly 1000 developers representing well over 100 corporations.*

*Since 2005, over 3700 individual developers from over 200 different companies have contributed to the kernel. The Linux kernel, thus, has become a common resource developed on a massive scale by companies which are fierce competitors in other areas.*



March 2008\*

## Introduction

The Linux kernel is the lowest level of software running on a Linux system. It is charged with managing the hardware, running user programs, and maintaining the overall security and integrity of the whole system. It is this kernel, which after its initial release by Linus Torvalds in 1991, jump-started the development of Linux as a whole. The kernel is a relatively small part of the software on a full Linux system (many other large components come from the GNU project, the GNOME and KDE desktop projects, the X.org project, and many other sources), but it is the core which determines how well the system will work and is the piece which is truly unique to Linux.

The Linux kernel is an interesting project to study for a number of reasons. It is one of the largest individual components on almost any Linux system. It also features one of the fastest-moving development processes and involves more developers than any other open source project. This paper looks at how that process works, focusing on nearly three years of kernel history as represented by the 2.6.11 through 2.6.24 releases.

## Development Model

With the 2.6.x series, the Linux kernel has moved to a relatively strict, time-based release model. At the 2005 Kernel Developer Summit in Ottawa, Canada, it was decided that kernel releases would happen every 2-3 months, with each release being a “major” release in that it includes new features and internal API changes.

The quick release cycle was chosen as a way to get new features out to users in a stable form with minimal delay. As a result, new code – features, device drivers, etc. – is available in a stable kernel within a few months of its completion, minimizing or eliminating the need for distributors to backport developmental code into stable releases. So the kernels released by distributors contain many fewer distribution-specific modifications, yielding higher stability and fewer differences between distributions.

Each 2.6.x release is a stable release, in that it is made available when the list of outstanding bugs is made as small as possible. For problems which turn up after a kernel release, the “-stable” branch exists as a way to quickly get fixes out to the community. This is best explained with the diagram shown in Figure 1.

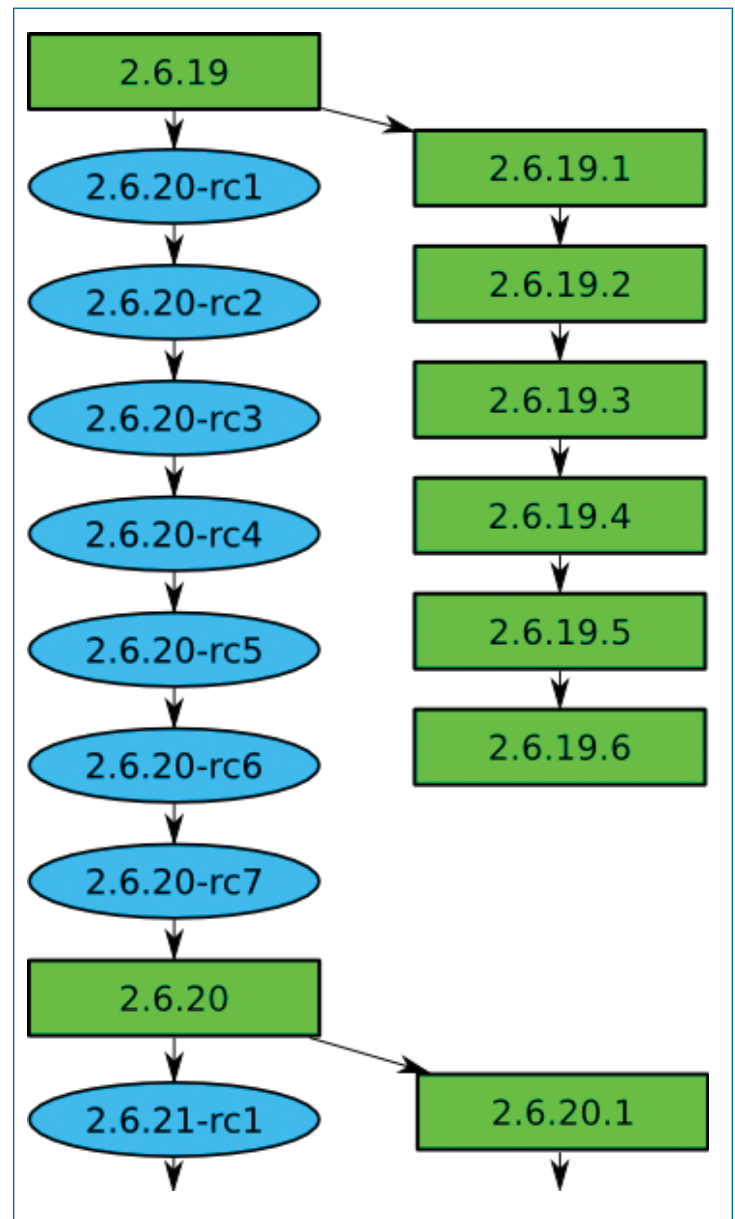


Figure 1 – Linux Kernel Release Cycle

The kernel team released the 2.6.19 kernel as a stable release. Then the developers started working on new features and started releasing the release candidate versions as development kernels so that people could help test and debug the changes. After everyone agreed that the development release was stable enough, it was released as the 2.6.20 kernel.

While the development of new features was happening, the 2.6.19.1, 2.6.19.2 and other stable kernel versions were released, containing bug fixes and security updates.

This paper focuses exclusively on the main 2.6.x releases, to the exclusion of the stable updates. Those updates are small, and, in any case, the design of the development process requires that fixes accepted for -stable also be accepted into the mainline for the next major release.

## Release Frequency

When the kernel developers first decided on this new development cycle, it was said that a new kernel would be released every 2-3 months, in order to prevent lots of new development from being “backed up.” The actual number of days between releases can be seen in Table 1.

Kernel Version	Release Date	Days of Development
2.6.11	2005-03-02	69
2.6.12	2005-05-17	108
2.6.13	2005-08-28	73
2.6.14	2005-10-27	61
2.6.15	2006-01-02	68
2.6.16	2006-03-19	77
2.6.17	2006-06-17	91
2.6.18	2006-09-19	95
2.6.19	2006-11-29	72
2.6.20	2007-02-04	68
2.6.21	2007-04-21	81
2.6.22	2007-07-08	75
2.6.23	2007-10-09	94
2.6.24	2008-01-24	108

Table 1 – Frequency of kernel releases

It turns out that they were very correct, with the average being 2.7 months between releases.

## Rate of Change

When preparing work for submission to the Linux kernel, developers break their changes down into small, individual units, called patches. These patches usually do only one thing to the source code; they are built on top of each other, modifying the source code by changing, adding, or removing lines of code. Each patch should, when applied, yield a kernel which still builds and works properly.

This discipline forces kernel developers to break their changes down into small, logical pieces; as a result, each change can be reviewed for code quality and correctness. One other result is that the number of individual changes that go into each kernel release is very large, as can be seen in Figure 2.

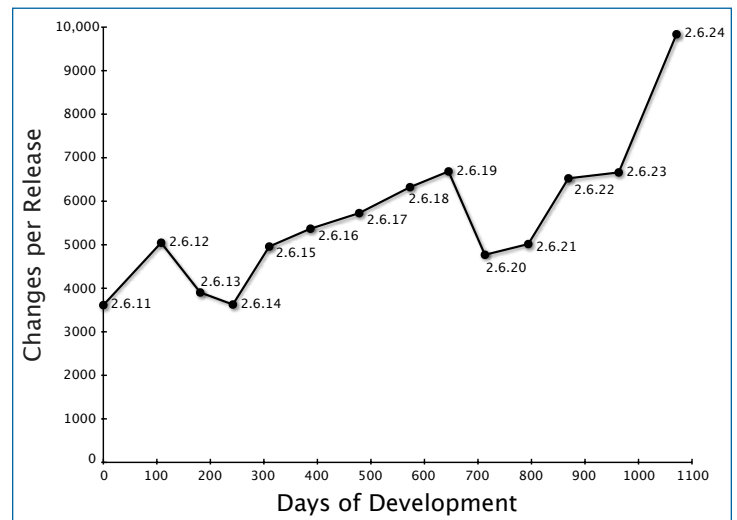


Figure 2 – Changes per kernel release

By taking into account the amount of time required for each kernel release, one can arrive at the number of changes accepted into the kernel per hour. The results can be seen in Figure 3.

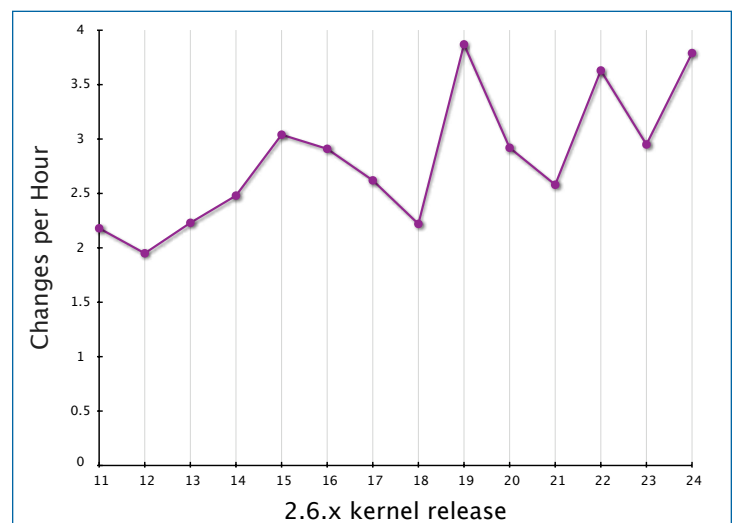


Figure 3 – Changes per hour by kernel release

So, from the 2.6.11 to the 2.6.24 kernel release (a total of 1140 days), there were, on average, 2.83 patches applied to the kernel tree per hour. And that is only the patches that were accepted. The ability to sustain this rate of change for years is unprecedented in any previous public software project.

## Kernel Source Size

The Linux kernel keeps growing in size over time as more hardware is supported and new features are added. For the following numbers, we have counted everything in the released Linux source

package as “source code” even though a small percentage of the total are the scripts used to configure and build the kernel, as well as a minor amount of documentation. Those files, too, are part of the larger work, and thus merit being counted.

The information in Figure 4 show the number of files and lines in each kernel version.

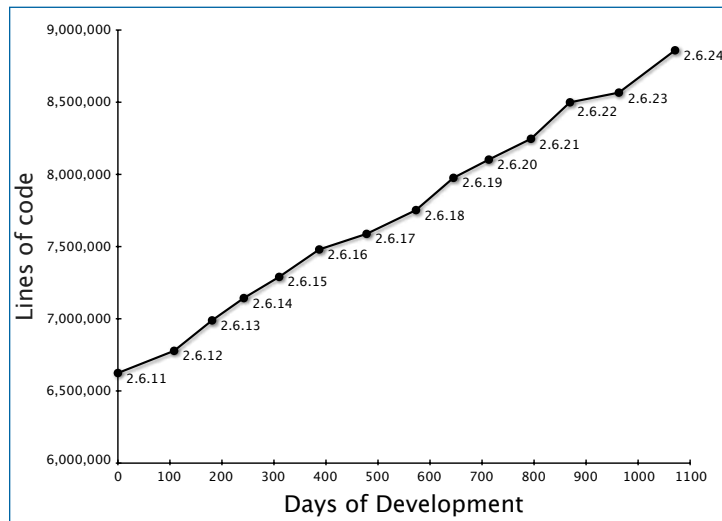


Figure 4 – Size per kernel release

Over these releases, the kernel team has a very constant growth rate of about 10% per year, a very impressive number given the size of the code tree. But the kernel is not just growing. With every change that is made to the kernel source tree, lines are added, modified, and deleted in order to accomplish the needed changes. Looking at these numbers, broken down by days, shows how quickly the kernel source tree is being worked on over time. This can be seen in Figure 5.

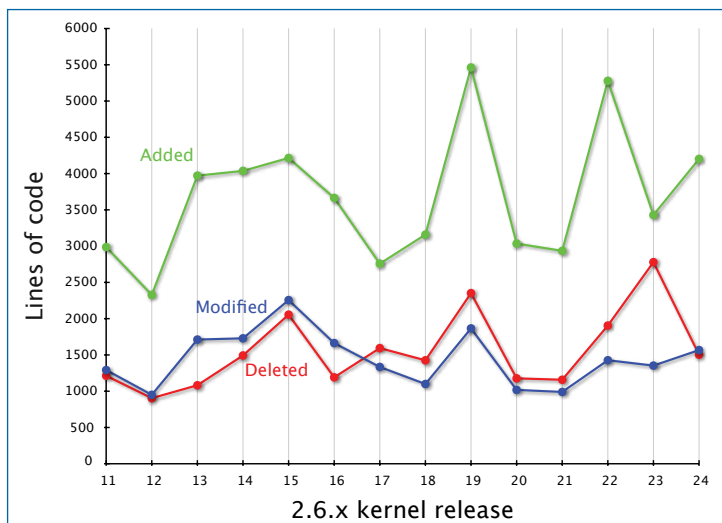


Figure 5 – Rate of change by kernel release

Summing up these numbers, it comes to an impressive 3,621 lines added, 1,550 lines removed, and 1,425 lines changed every day for the past 2 1/2 years. That rate of change is larger than any other public software project of any size.

## Who is Doing the Work

The number of different developers who are doing Linux kernel development and the identifiable companies<sup>1</sup> who are sponsoring this work, have been increasing over the different kernel versions, as can be seen in Table 2.

Kernel Version	# of Developers	# of Known Companies
2.6.11	483	71
2.6.12	701	90
2.6.13	637	91
2.6.14	625	89
2.6.15	679	96
2.6.16	775	100
2.6.17	784	106
2.6.18	897	121
2.6.19	878	126
2.6.20	728	130
2.6.21	834	132
2.6.22	957	176
2.6.23	991	178
2.6.24	1,057	186
All	3,678	271

Table 2 – Number of individual developers and employers

In fact, the individual development community has doubled in the last three years.

Despite the large number of individual developers, there is still a relatively small number who are doing the majority of the work. Over the past three years, the top 10 individual developers have contributed almost 15 percent of the number of changes and the top 30 developers have contributed 30 percent. The list of individual developers, the number of changes they have contributed, and the percentage of the overall total can be seen in Table 3.

<sup>1</sup> The identification of the different companies is described in the next section.

Name	# of Changes	% of Total Changes
Al Viro	1571	1.9%
David S. Miller	1520	1.8%
Adrian Bunk	1441	1.7%
Ralf Baechle	1346	1.6%
Andrew Morton	1222	1.5%
Andi Kleen	993	1.2%
Takashi Iwai	963	1.2%
Tejun Heo	938	1.1%
Russell King	926	1.1%
Stephen Hemminger	920	1.1%
Thomas Gleixner	754	0.9%
Patrick McHardy	740	0.9%
Ingo Molnar	735	0.9%
Trond Myklebust	664	0.8%
Neil Brown	646	0.8%
Randy Dunlap	645	0.8%
Jean Delvare	617	0.7%
Jeff Garzik	615	0.7%
Christoph Hellwig	615	0.7%
David Brownell	588	0.7%
Paul Mundt	581	0.7%
Alan Cox	571	0.7%
Jeff Dike	558	0.7%
Herbert Xu	538	0.6%
David Woodhouse	503	0.6%
Greg Kroah-Hartman	496	0.6%
Linus Torvalds	495	0.6%
Dmitry Torokhov	494	0.6%
Alan Stern	478	0.6%
Ben Dooks	477	0.6%

Table 3 – Individual kernel contributors

## Who is Sponsoring the Work

The Linux kernel is a resource which is used by a large variety of companies. Many of those companies never participate in the development of the kernel; they are content with the software as it is and do not feel the need to help drive its development in any particular direction. But, as can be seen in Table 4, an increasing number of companies are working toward the improvement of the kernel.

Company Name	# of Changes	% of Total
None	11,594	13.9%
Unknown	10,803	12.9%
Red Hat	9,351	11.2%
Novell	7,385	8.9%
IBM	6,952	8.3%
Intel	3,388	4.1%
Linux Foundation	2,160	2.6%
Consultant	2,055	2.5%
SGI	1,649	2.0%
MIPS Technologies	1,341	1.6%
Oracle	1,122	1.3%
MontaVista	1,010	1.2%
Google	965	1.1%
Linutronix	817	1.0%
HP	765	0.9%
NetApp	764	0.9%
SWsoft	762	0.9%
Renesas Technology	759	0.9%
Freescale	730	0.9%
Astaro	715	0.9%
Academia	656	0.8%
Cisco	442	0.5%
Simtec	437	0.5%
Linux Networx	434	0.5%
QLogic	398	0.5%
Fujitsu	389	0.5%
Broadcom	385	0.5%
Analog Devices	358	0.4%
Mandriva	329	0.4%
Mellanox	294	0.4%
Snapgear	285	0.3%

Table 4 – Companies working toward the improvement of the kernel

Below we look more closely at the companies which are employing kernel developers. For each developer, corporate affiliation was obtained through one or more of the following: (1) the use of company email addresses, (2) sponsorship information included in the code they submit, or (3) simply asking the developers directly. The numbers presented are necessarily approximate; developers occasionally change employers, and they may do personal work out of the office. But they will be close enough to support a number of conclusions.

There are a number of developers for whom we were unable to determine a corporate affiliation; those are grouped under “unknown” in Table 4. With few exceptions, all of the people in this category have contributed 10 or fewer changes to the kernel over the past three years, yet the large number of these developers causes their total contribution to be quite high.

The category “None,” instead, represents developers who are known to be doing this work on their own, with no financial contribution happening from any company.

The top 10 contributors, including the groups “unknown” and “none” make up over 75% of the total contributions to the kernel. It is worth noting that, even if one assumes that all of the “unknown” contributors were working on their own time, over 70% of all kernel development is demonstrably done by developers who are being paid for their work.

What we see here is that a small number of companies are responsible for a large portion of the total changes to the kernel. But there is a “long tail” of companies which have made significant changes. There may be no other examples of such a large, common resource being supported by such a large group of independent actors in such a collaborative way.

### Why Companies Support Kernel Development

The list of companies participating in Linux kernel development includes many of the most successful technology firms in existence. None of these companies are supporting Linux development as an act of charity; in each case, these companies find that improving the kernel helps them to be more competitive in their markets. Some examples:

- Companies like IBM, Intel, SGI, MIPS, Freescale, HP, etc. are all working to ensure that Linux runs well on their hardware. That, in turn, makes their offerings more attractive to Linux users, resulting in increased sales.
- Distributors like Red Hat, Novell, and MontaVista have a clear interest in making Linux as capable as it can be. Though these firms compete strongly with each other for customers, they all work together to make the Linux kernel better.
- Companies like Sony, Nokia, and Samsung ship Linux as a component of products like video cameras, television sets, and mobile telephones. Working with the development process helps these companies ensure that Linux will continue to be a solid base for their products in the future.
- Companies which are not in the information technology business can still find working with Linux beneficial. The 2.6.25 kernel will

include an implementation of the PF\_CAN network protocol which was contributed by Volkswagen. PF\_CAN allows for reliable communications between components in an interference-prone environment – such as that found in an automobile. Linux gave Volkswagen a platform upon which it could build its networking code; the company then found it worthwhile to contribute the code back so that it could be maintained with the rest of the kernel. See <http://lwn.net/Articles/253425/> for more information on this work.

There are a number of good reasons for companies to support the Linux kernel. As a result, Linux has a broad base of support which is not dependent on any single company. Even if the largest contributor were to cease participation tomorrow, the Linux kernel would remain on a solid footing with a large and active development community.

### Conclusion

The Linux kernel is one of the largest and most successful open source projects that has ever come about. The huge rate of change and number of individual contributors shows that it has a vibrant and active community, constantly causing the evolution of the kernel in response to a number of different environments it is used in. There are enough companies participating to fund the bulk of the development effort, even if many companies which could benefit from contributing to Linux have, thus far, chosen not to. With the current expansion of Linux in the server, desktop and embedded markets, it's reasonable to expect this number of contributing companies – and individual developers – will continue to increase.

### Thanks

The authors would like to thank the thousands of individual kernel contributors; without them, papers like this would not be interesting to anyone.

### Resources

We would also like to acknowledge Jonathan Corbet's gitdm tool that was used to create a large number of these different statistics. The information for this paper was retrieved directly from the Linux kernel releases as found at the kernel.org web site and from the git kernel repository. Some of the logs from the git repository were cleaned up by hand due to email addresses changing over time, and minor typos in authorship information. A spreadsheet was used to compute a number of the statistics. All of the logs, scripts, and spreadsheets can be found at [http://www.kernel.org/pub/linux/kernel/people/gregkh/kernel\\_history](http://www.kernel.org/pub/linux/kernel/people/gregkh/kernel_history).

\* Based on a paper originally published at the 2006 Linux Symposium