Linux Kernel Development

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

Greg Kroah-Hartman
SuSE Labs / Novell Inc.
gregkh@suse.de

1 Introduction

The Linux kernel is one of the most popular Open Source development projects, and yet not much attention has been placed on who is doing this development, who is sponsoring this development, and what exactly is being developed. This paper should help explain some of these facts by delving into the kernel changelogs and producing lots of statistics.

This paper will focus on the kernel releases of the past two and 1/3 years, from the 2.6.11 through the 2.6.21 release.

2 Development vs. Stability

In the past, the Linux kernel was split into two different trees, the *development* branch, and the *stable* branch. The development branch was specified by using an odd number for the second release number, while the stable branch used an even number. As an example, the 2.5.32 release was a development release, while the 2.4.24 release is a stable release.

After the 2.6 kernel series was created, the developers decided to change this method of having two different trees. They declared that all 2.6 kernel releases would be considered "stable," no matter how quickly development was happening. These releases would happen every 2 to 3 months and would allow developers to add new features and then stabilize them in time for the next release. This was done in order to allow distributions to be able to decide on a release point easier by always having at least one stable kernel release near a distribution release date.

To help with stability issues while the developers are creating a new kernel version, a `-stable` branch was created that would contain bug fixes and security updates for the past kernel release before the next major release happened.

This is best explained with the diagram shown in Figure 1. 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 `-rc` 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.

For this paper, we are going to focus on the main kernel releases, and ignore the `-stable` releases, as they contain a very small number of bugfixes and are not where any development happens.

3 Frequency of release

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.

It turns out that they were very correct, with the average being 2.6 months between releases.

4 Rate of Change

When modifying the Linux kernel, developers break their changes down into small, individual units of change, called patches. These patches usually do only

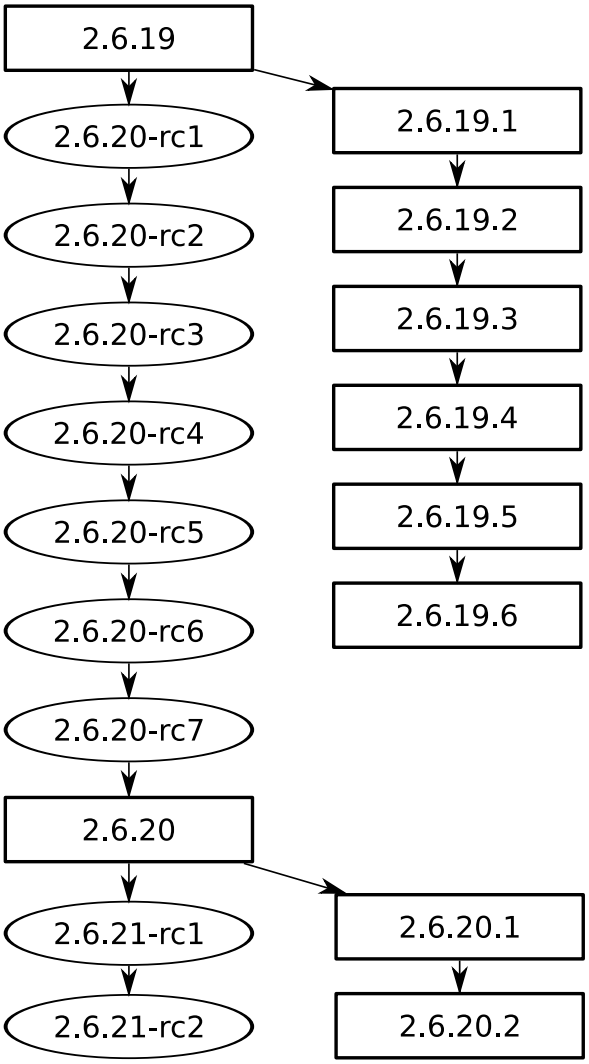

Figure 1: Kernel release cycles

Kernel Version	Days of development
2.6.11	69
2.6.12	108
2.6.13	73
2.6.14	61
2.6.15	68
2.6.16	77
2.6.17	91
2.6.18	95
2.6.19	72
2.6.20	68
2.6.21	81

Table 1: Frequency of kernel releases

Kernel Version	Changes per Release
2.6.11	4,041
2.6.12	5,565
2.6.13	4,174
2.6.14	3,931
2.6.15	5,410
2.6.16	5,734
2.6.17	6,113
2.6.18	6,791
2.6.19	7,073
2.6.20	4,983
2.6.21	5,349

Table 2: Changes per kernel release

one thing to the source tree, and are built on top of each other, modifying the source code by changing, adding, or removing lines of code. At each change point in time, the kernel should be able to be successfully built and operate. By enforcing this kind of discipline, the kernel developers must break their changes down into small logical pieces. The number of individual changes that go into each kernel release is very large, as can be seen in Table 2.

When you compare the number of changes per release, with the length of time for each release, you can determine the number of changes per hour, as can be seen in Table 3.

So, from the 2.6.11 to the 2.6.21 kernel release, a total of 852 days, there were 2.89 patches applied to the kernel

Kernel Version	Changes per Hour
2.6.11	2.44
2.6.12	2.15
2.6.13	2.38
2.6.14	2.69
2.6.15	3.31
2.6.16	3.10
2.6.17	2.80
2.6.18	2.98
2.6.19	4.09
2.6.20	3.05
2.6.21	2.75

Table 3: Changes per hour by kernel release

tree per hour. And that is only the patches that were accepted.

5 Kernel Source Size

The Linux kernel keeps growing in size over time, as more hardware is supported, and new features added. For the following numbers, I count everything in the released Linux source tarball as "source code" even though a small percentage is the scripts used to configure and build the kernel, as well as a minor amount of documentation. This is done because someone creates those files, and are worthy of being mentioned.

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

Kernel Version	Files	Lines
2.6.11	17,091	6,624,076
2.6.12	17,361	6,777,860
2.6.13	18,091	6,988,800
2.6.14	18,435	7,143,233
2.6.15	18,812	7,290,070
2.6.16	19,252	7,480,062
2.6.17	19,554	7,588,014
2.6.18	20,209	7,752,846
2.6.19	20,937	7,976,221
2.6.20	21,281	8,102,533
2.6.21	21,615	8,246,517

Table 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.

When you combine the number of lines added per release, and compare it to the amount of time per release, you can get some very impressive numbers, as can be seen in Table 5.

Summing up these numbers, it comes to a crazy 85.63 new lines of code being added to the kernel tree every hour for the past 2 1/3 years.

6 Where the Change is Happening

The Linux kernel source tree is highly modular, enabling new drivers and new architectures to be added quite easily. The source code can be broken down into the following categories:

Kernel Version	Lines per Hour
2.6.11	77.6
2.6.12	59.3
2.6.13	120.4
2.6.14	105.5
2.6.15	90.0
2.6.16	102.8
2.6.17	49.4
2.6.18	72.3
2.6.19	129.3
2.6.20	77.4
2.6.21	74.1

Table 5: Lines per hour by kernel release

- **core**: this is the core kernel code, run by everyone and included in all architectures. This code is located in the subdirectories `block/`, `ipc/`, `init/`, `kernel/`, `lib/`, `mm/`, and portions of the `include/` directory.
- **drivers**: these are the drivers for different hardware and virtual devices. This code is located in the subdirectories `crypto/`, `drivers/`, `sound/`, `security/`, and portions of the `include/` directory.
- **architecture**: this is the CPU specific code, where anything that is only for a specific processor lives. This code is located in the `arch/`, and portions of the `include/` directory.
- **network**: this is the code that controls the different networking protocols. It is located in the `net/` directory and the `include/net` subdirectory.
- **filesystems**: this is the code that controls the different filesystems. It is located in the `fs/` directory.
- **miscellaneous**: this is the rest of the kernel source code, including the code needed to build the kernel, and the documentation for various things. It is located in `Documentation/`, `scripts/`, and `usr/` directories.

The breakdown of the 2.6.21 kernel's source tree by the number of different files in the different category is shown in Table 6, while Table 7 shows the breakdown by the number of lines of code.

Category	Files	% of kernel
core	1,371	6%
drivers	6,537	30%
architecture	10,235	47%
network	1,095	5%
filesystems	1,299	6%
miscellaneous	1,068	5%

Table 6: 2.6.21 Kernel size by files

Category	Lines of Code	% of kernel
core	330,637	4%
drivers	4,304,859	52%
architecture	2,127,154	26%
network	506,966	6%
filesystems	702,913	9%
miscellaneous	263,848	3%

Table 7: 2.6.21 Kernel size by lines of code

In the 2.6.21 kernel release, the architecture section of the kernel contains the majority of the different files, but the majority of the different lines of code are by far in the drivers section.

I tried to categorize what portions of the kernel are changing over time, but there did not seem to be a simple way to represent the different sections changing based on kernel versions. Overall, the percentage of change seemed to be evenly spread based on the percentage that the category took up within the overall kernel structure.

7 Who is Doing the Work

The number of different developers who are doing Linux kernel development, and the identifiable companies[1] who are sponsoring this work, have been slowly increasing over the different kernel versions, as can be seen in Table 8.

[1]The identification of the different companies is described in the next section.

Kernel Version	Number of Developers	Number of Companies
2.6.11	479	30
2.6.12	704	38
2.6.13	641	39
2.6.14	632	45
2.6.15	685	49
2.6.16	782	56
2.6.17	787	54
2.6.18	904	60
2.6.19	887	67
2.6.20	730	75
2.6.21	838	68
All	2998	83

Table 8: Number of individual developers and employers

Factoring in the amount of time between each individual kernel releases and the number of developers and employers ends up showing that there really is an increase of the size of the community, as can be shown in Table 9.

Despite this large number of individual developers, there is still a small number who are doing the majority of the work. Over the past two and one half years, the top 10 individual developers have contributed 15 percent of the number of changes and the top 30 developers

Kernel Version	Number of Developers per day	Number of Companies per day
2.6.11	6.94	0.43
2.6.12	6.52	0.35
2.6.13	8.78	0.53
2.6.14	10.36	0.74
2.6.15	10.07	0.72
2.6.16	10.16	0.73
2.6.17	8.65	0.59
2.6.18	9.52	0.63
2.6.19	12.32	0.93
2.6.20	10.74	1.10
2.6.21	10.35	0.84

Table 9: Number of individual developers and employers over time

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 10.

Name	Number of Changes	Percent of Total
Al Viro	1326	2.2%
David S. Miller	1096	1.9%
Adrian Bunk	1091	1.8%
Andrew Morton	991	1.7%
Ralf Baechle	981	1.7%
Andi Kleen	856	1.4%
Russell King	788	1.3%
Takashi Iwai	764	1.3%
Stephen Hemminger	650	1.1%
Neil Brown	626	1.1%
Tejun Heo	606	1.0%
Patrick McHardy	529	0.9%
Randy Dunlap	486	0.8%
Jaroslav Kysela	463	0.8%
Trond Myklebust	445	0.8%
Jean Delvare	436	0.7%
Christoph Hellwig	435	0.7%
Linus Torvalds	433	0.7%
Ingo Molnar	429	0.7%
Jeff Garzik	424	0.7%
David Woodhouse	413	0.7%
Paul Mackerras	411	0.7%
David Brownell	398	0.7%
Jeff Dike	397	0.7%
Ben Dooks	392	0.7%
Greg Kroah-Hartman	388	0.7%
Herbert Xu	376	0.6%
Dave Jones	371	0.6%
Ben Herrenschmidt	365	0.6%
Mauro Chehab	365	0.6%

Table 10: Individual Kernel contributors

8 Who is Sponsoring the Work

Despite the broad use of the Linux kernel in a wide range of different types of devices, and reliance of it by a number of different companies, the number of individual companies that help sponsor the development of the Linux kernel remains quite small as can be seen by the list of different companies for each kernel version in Table 8.

The identification of the different companies was deduced through the use of company email addresses and the known sponsoring of some developers. It is possible that a small number of different companies were missed, however based on the analysis of the top contributors of the kernel, the majority of the contributions are attributed in this paper.

The large majority of contributions still come from individual contributors, either because they are students, they are contributing on their own time, or their employers are not allowing them to use their company email addresses for their kernel development efforts. As seen in Table 11 almost half of the contributions are done by these individuals.

Company Name	Number of Changes	Percent of Total
Unknown	27976	47.3%
Red Hat	6106	10.3%
Novell	5923	10.0%
Linux Foundation	4843	8.2%
IBM	3991	6.7%
Intel	2244	3.8%
SGI	1353	2.3%
NetApp	636	1.1%
Freescale	454	0.8%
linutronix	370	0.6%
HP	360	0.6%
Harvard	345	0.6%
SteelEye	333	0.6%
Oracle	319	0.5%
Conectiva	296	0.5%
MontaVista	291	0.5%
Broadcom	285	0.5%
Fujitsu	266	0.4%
Veritas	219	0.4%
QLogic	218	0.4%
Snapgear	214	0.4%
Emulex	147	0.2%
LSI Logic	130	0.2%
SANPeople	124	0.2%
Qumranet	106	0.2%
Atmel	91	0.2%
Toshiba	90	0.2%
Samsung	82	0.1%
Renesas Technology	81	0.1%
VMWare	78	0.1%

Table 11: Company Kernel Contributions

9 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 show that it has a vibrant and active community, constantly causing the evolution of the kernel to survive the number of different environments it is used in. However, despite the large number of individual contributors, the sponsorship of these developers seem to be consolidated in a small number of individual companies. It will be interesting to see if, over time, the companies that rely on the success of the Linux kernel will start to sponsor the direct development of the project, to help ensure that it remains valuable to those companies.

10 Thanks

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

I would also like to thank Jonathan Corbet, whose `gitdm` tool were used to create a large number of these different statistics. Without his help, this paper would have taken even longer to write, and not been as informative.

11 Resources

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 spreadsheet can be found at `http://www.kernel.org/pub/linux/kernel/people/gregkh/kernel_history/`

Proceedings of the Linux Symposium

Volume One

June 27th–30th, 2007
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*
C. Craig Ross, *Linux Symposium*

Review Committee

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*
Dirk Hohndel, *Intel*
Martin Bligh, *Google*
Gerrit Huizenga, *IBM*
Dave Jones, *Red Hat, Inc.*
C. Craig Ross, *Linux Symposium*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*
Gurhan Ozen, *Red Hat, Inc.*
John Feeney, *Red Hat, Inc.*
Len DiMaggio, *Red Hat, Inc.*
John Poelstra, *Red Hat, Inc.*