» The Linux Foundation

Linux Kernel Development

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

December 2010

Jonathan Corbet, LWN.net
Greg Kroah-Hartman, SuSE Labs / Novell Inc.
Amanda McPherson, The Linux Foundation

A White Paper By The Linux Foundation
http://www.linuxfoundation.org

Summary

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 approximately 10,000 patches going into each recent kernel release. These releases each contain the work of over 1000 developers representing around 200 corporations.

Since 2005, over 6100 individual developers from over 600 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.

The first version of this study was published in 2008; it was then updated in 2009. That update noted a number of changes, including a 10% increase in the number of developers participating in each release cycle, a notable increase in the number of companies participating, and a tripling of the rate at which code is being added to the kernel. At that time, all of the numbers were in a period of rapid increase.

As documented in the last paper, in 2009 the Linux community saw, with the release of 2.6.30, a peak in the lines of code added. This can largely be attributed to significant new features being added to the kernel, most notably the first additions of Btrfs, perf and ftrace, as well as the peak of the inflow from the Linux-staging tree that had been happening for some time.

This update shows a slightly different picture. The number of commits peaked with the 2.6.30 release; the number of commits for 2.6.35 was 18% lower. Most other metrics have fallen as well.

In short, we see a step back from the frenzied activity of 2.6.30 even though the number of developers involved has fallen only slightly since its peak in 2.6.32.

The numbers in this edition of the paper reflect the natural development cycle of an operating system that had major pieces added/changed in the previous year. Of course the Linux kernel community is still hard at work and growing. In fact, there have been 1.5 million lines of code added to the kernel since the 2009 update. Since the last paper, additions and changes translate to an amazing 9,058 lines added, 4,495 lines removed, and 1,978 lines changed every day - weekends and holidays included.

The data in this year's update also shows a good showing of new players in the Linux kernel development space from the world of mobile/consumer electronics and embedded technology (and their suppliers). This is a healthy development and not surprising given the growth of Linux usage in embedded devices, even though the authors would like to see more companies from that space participate in the Linux development community.

The overall picture shows a robust development community which continues to grow both in size and in productivity.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

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. Since 2005, kernel development history is also quite well documented, thanks to the use of the Git source code management system.

This paper takes advantage of that development history to look at how the process works, focusing on over five years of kernel history as represented by the 2.6.11 through 2.6.35 releases. This is the third version of this paper, following up on http://www.linuxfoundation.org/sites/main/files/publications/linuxkerneldevelopment.pdf the original study which was published in April, 2008, and http://www.linuxfoundation.org/sites/main/files/publications/whowriteslinux.pdf the 2009 update, which looked at the history through the 2.6.30 release. A look at the five kernel releases which have happened since then shows that, while many things remain the same, others are changing.

Development Model

Linux kernel development proceeds under a loose, time-based release model, with a new major kernel release occurring every 2-3 months. This model, which was first formalized in 2005, gets new features into the mainline kernel and out to users with a minimum of delay. That, in turn, speeds the pace of development and minimizes the number of external changes that distributors need to apply. As a result, distributor kernels contain relatively few distribution-specific changes; this leads to higher quality and fewer differences between distributions.

One significant change since the initial version of this paper is the establishment of the linux-next tree. Linux-next serves as a staging area for the next kernel development cycle; as of this writing, 2.6.36 is in the stabilization phase, so linux-next contains changes intended for 2.6.37. This repository gives developers a better view of which changes are coming in the future and helps them to ensure that there will be a minimum of integration problems when the next development cycle begins. Linux-next smooths out the development cycle, helping it to scale to higher rates of change.

After each mainline 2.6 release, the kernel's "stable team" (currently Greg Kroah-Hartman) takes up short-term maintenance, applying important fixes as they are developed. The stable process

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

ensures that important fixes are made available to distributors and users and that they are incorporated into future mainline releases as well. The stable maintenance period lasts a minimum of one development cycle and, for specific kernel releases, can go significantly longer; some stable update statistics will be provided below.

Release Frequency

The desired release period for a major kernel release is, by common consensus, 8-12 weeks. A much-shorter period would not give testers enough times to find problems with new kernels, while a longer period would allow too much work to pile up between releases. The actual time between kernel releases tends to vary a bit, depending on the size of the release and the difficulty encountered in tracking down the last regressions. Since 2.6.11, the actual kernel release history looks like:

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	2007-04-25	81
2.6.22	2007-07-08	75
2.6.23	2007-10-09	94
2.6.24	2008-01-24	108
2.6.25	2008-04-16	83
2.6.26	2008-07-13	88
2.6.27	2008-10-09	88
2.6.28	2008-12-24	76
2.6.29	2009-03-23	89
2.6.30	2009-06-09	78
2.6.31	2009-09-09	92
2.6.32	2009-12-02	84
2.6.33	2010-02-24	84
2.6.34	2010-05-15	81
2.6.35	2010-08-01	77

The average kernel development cycle currently runs for 81 days, just under twelve weeks.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

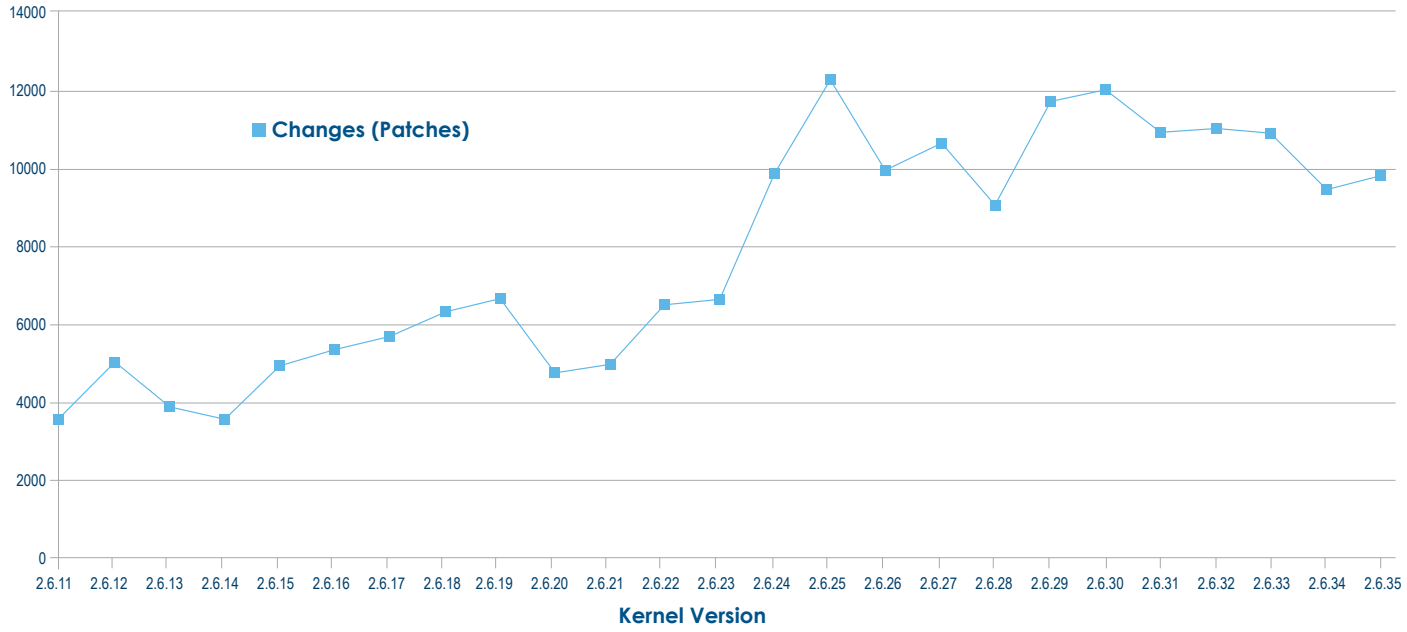
Rate of Change

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 the table below:

Kernel Version	Changes (Patches)
2.6.11	3,616
2.6.12	5,047
2.6.13	3,904
2.6.14	3,627
2.6.15	4,959
2.6.16	5,369
2.6.17	5,727
2.6.18	6,323
2.6.19	6,685
2.6.20	4,768
2.6.21	5,016
2.6.22	6,526
2.6.23	6,662
2.6.24	9,836
2.6.25	12,243
2.6.26	9,941
2.6.27	10,628
2.6.28	9,048
2.6.29	11,678
2.6.30	11,989
2.6.31	10,883
2.6.32	10,989
2.6.33	10,871
2.6.34	9,443
2.6.35	9,801

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

Changes to the Kernel Over Time



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 this table:

Kernel Version	Changes Per Hour
2.6.11	2.18
2.6.12	1.95
2.6.13	2.23
2.6.14	2.48
2.6.15	3.04
2.6.16	2.91
2.6.17	2.62
2.6.18	2.22
2.6.19	3.87
2.6.20	2.92
2.6.21	2.58
2.6.22	3.63
2.6.23	2.95
2.6.24	3.79
2.6.25	6.15
2.6.26	4.71
2.6.27	5.03
2.6.28	4.96
2.6.29	5.47

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

Kernel Version	Changes Per Hour
2.6.30	6.40
2.6.31	4.93
2.6.32	5.46
2.6.33	5.39
2.6.34	4.86
2.6.35	5.30

So, between the 2.6.11 and 2.6.35 kernel releases (which were 1902 days apart), there were, on average, 4.02 patches applied to the kernel tree per hour. In the time since the publication of the previous version of this paper, that rate has been significantly higher: 5.18 patches per hour. As the Linux kernel grows, the rate of change is growing with it.

The rate of change has slowed slightly from the rate (5.45 patches/hour) reported in the 2009 update; the peak rate seen with the 2.6.30 kernel release has not been repeated. Development rates are naturally variable, and the rates for the kernel have never increased in a monotonic fashion; that said, the rate of change has remained notably lower for the last year. There are a couple of explanations for that trend:

- The kernels since 2.6.30 have seen the completion and stabilization of a number of long-term projects, including the ext4 and btrfs filesystems, the addition of the ftrace and perf events subsystems, and the reimplementation of our graphics layer. Rates of change will naturally slow as the finishing touches are put on these developments.
- The addition of the staging tree in 2.6.28 began a process of merging a large amount of out-of-tree code into the mainline kernel. By the 2.6.31 development cycle, that process was slowing down as the backlog of code was taken care of. There are still new drivers entering the kernel via the staging tree, but they are now arriving at a rate which more closely reflects the actual rate of development.

The burst of activity caused by the staging tree is not likely to be repeated anytime soon, but the pace of kernel development as a whole can be expected to increase as developers take on new challenges in the future.

It is also worth noting that the above figures understate the total level of activity; most patches go through a number of revisions before being accepted into the mainline kernel, and many are never accepted at all. The ability to sustain this rate of change for years is unprecedented in any previous public software project.

Stable Updates

As mentioned toward the beginning of this document, kernel development does not stop with a mainline release. Inevitably, problems will be found in released kernels, and patches will be made to fix those problems. The stable kernel update process was designed to capture those patches in a way that ensures that both the mainline kernel and current releases are fixed. These stable

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

updates are the base from which most distributor kernels are made.

The stable kernel update history (since the stable kernel process was introduced after the 2.6.11 release) looks like this:

Kernel Version	Total Updates	Fixes
2.6.11	12	79
2.6.12	6	53
2.6.13	5	44
2.6.14	7	96
2.6.15	7	110
2.6.16	62	1053
2.6.17	14	191
2.6.18	8	240
2.6.19	7	189
2.6.20	21	447
2.6.21	7	162
2.6.22	19	379
2.6.23	16	302
2.6.24	7	246
2.6.25	20	492
2.6.26	8	321
2.6.27	53	1553
2.6.28	10	613
2.6.29	6	383
2.6.30	10	419
2.6.31	14	826
2.6.32	21	1793
2.6.33	7	883
2.6.34	7	601
2.6.35	4	228

As can be seen, the number of updates going into stable kernels has grown over the years. The main driver for this increase is a much higher level of discipline in the development community: we have gotten much better at evaluating patches and identifying those which are applicable to releaased kernels. Additionally, some kernels are receiving stable updates for relatively long periods of time; the 2.6.27 kernel is still being updated as of this writing.

With just over five years of history, the stable update series has proven its value by allowing the final fixes to be made to released kernels while, simultaneously, letting mainline development move forward.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

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 is 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 the following table shows the number of files and lines in each kernel version.

Kernel Version	Files	Lines
2.6.11	17,090	6,624,076
2.6.12	17,360	6,777,860
2.6.13	18,090	6,988,800
2.6.14	18,434	7,143,233
2.6.15	18,811	7,290,070
2.6.16	19,251	7,480,062
2.6.17	19,553	7,588,014
2.6.18	20,208	7,752,846
2.6.19	20,936	7,976,221
2.6.20	21,280	8,102,533
2.6.21	21,614	8,246,517
2.6.22	22,411	8,499,410
2.6.23	22,530	8,566,606
2.6.24	23,062	8,859,683
2.6.25	23,813	9,232,592
2.6.26	24,273	9,411,841
2.6.27	24,356	9,630,074
2.6.28	25,276	10,118,757
2.6.29	26,702	10,934,554
2.6.30	27,911	11,560,971
2.6.31	29,143	11,970,124
2.6.32	30,504	12,532,677
2.6.33	31,584	12,912,684
2.6.34	32,316	13,243,582
2.6.35	33,335	13,468,253

Since the first version of this paper, the kernel has grown by almost 6.7 million lines of code - 1.5 million since the 2009 update. 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 the following table:

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

Kernel Version	Lines Added Per Day	Lines Deleted Per Day	Lines Modified Per Day
2.6.11	3,224	1,360	1,290
2.6.12	2,375	951	949
2.6.13	4,443	1,553	1,711
2.6.14	4,181	1,637	1,726
2.6.15	5,614	3,454	2,219
2.6.16	3,853	1,388	1,649
2.6.17	3,635	2,469	1,329
2.6.18	3,230	1,497	1,096
2.6.19	6,013	2,900	1,862
2.6.20	3,120	1,342	1,013
2.6.21	3,256	1,479	982
2.6.22	6,067	2,694	1,523
2.6.23	3,747	3,034	1,343
2.6.24	6,893	4,181	1,563
2.6.25	7,980	3,488	2,430
2.6.26	5,698	3,662	1,815
2.6.27	12,270	9,791	2,102
2.6.28	12,105	5,707	1,850
2.6.29	14,678	5,516	2,454
2.6.30	12,993	4,958	2,830
2.6.31	9,408	4,962	1,635
2.6.32	12,086	5,388	2,387
2.6.33	8,925	4,379	2,841
2.6.34	6,667	2,580	1,568
2.6.35	7,896	5,037	1,802

Summing up these numbers, it comes to an impressive 6,683 lines added, 3,774 lines removed, and 1,797 lines changed every day for the past 5.5 years. Since 2.6.30, those numbers jump to an amazing 9,058 lines added, 4,495 lines removed, and 1,978 lines changed every day - weekends and holidays included. 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 who are sponsoring this work, have been increasing over the different kernel versions, as can be seen in the following table. In fact, the individual development community has doubled in the last three years.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

Kernel Version	Number of Developers	Number of Known Companies
2.6.11	389	68
2.6.12	566	90
2.6.13	545	94
2.6.14	553	90
2.6.15	612	108
2.6.16	709	111
2.6.17	726	120
2.6.18	815	133
2.6.19	801	128
2.6.20	673	138
2.6.21	767	143
2.6.22	870	180
2.6.23	912	181
2.6.24	1,057	193
2.6.25	1,123	232
2.6.26	1,027	203
2.6.27	1,021	187
2.6.28	1,075	212
2.6.29	1,180	233
2.6.30	1,150	245
2.6.31	1,166	221
2.6.32	1,248	259
2.6.33	1,196	226
2.6.34	1,150	195
2.6.35	1,187	184
All	6,117	659

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

These numbers show a steady increase in the number of developers contributing to each kernel release over a period of several years.

Despite the large number of individual developers, there is still a relatively small number who are doing the majority of the work. In any given development cycle, approximately 1/3 of the developers involved contribute exactly one patch. Over the past 5.5 years, the top 10 individual developers have contributed 10% of the total changes and the top 30 developers have contributed almost 22% of the total. The list of individual developers, the number of changes they have contributed, and the percentage of the overall total can be seen here:

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

Name	Number of Changes	Percent of Changes
David S. Miller	2,533	1.3%
Ingo Molnar	2,273	1.2%
Al Viro	2,238	1.2%
Takashi Iwai	2,120	1.1%
Bartlomiej Zolnierkiewicz	2,014	1.1%
Adrian Bunk	1,918	1.0%
Paul Mundt	1,793	1.0%
Tejun Heo	1,691	0.9%
Ralf Baechle	1,577	0.8%
Greg Kroah-Hartman	1,506	0.8%
Andrew Morton	1,473	0.8%
Alan Cox	1,455	0.8%
Russell King	1,443	0.8%
Thomas Gleixner	1,389	0.7%
Mauro Carvalho Chehab	1,381	0.7%
Johannes Berg	1,334	0.7%
Christoph Hellwig	1,247	0.7%
Ben Dooks	1,214	0.6%
Patrick McHardy	1,205	0.6%
Andi Kleen	1,183	0.6%
Jean Delvare	1,180	0.6%
Randy Dunlap	1,171	0.6%
Trond Myklebust	1,022	0.5%
Stephen Hemminger	1,004	0.5%
Hans Verkuil	999	0.5%
Herbert Xu	969	0.5%
David Woodhouse	967	0.5%
Peter Zijlstra	951	0.5%
Alexey Dobriyan	949	0.5%
David Brownell	901	0.5%

The above numbers are drawn from the entire git repository history, starting with 2.6.12. If we look at the commits since the second version of this paper (2.6.30) through 2.6.35, the picture is similar but not identical:

Name	Number of Changes	Percent of Changes
Paul Mundt	665	1.3%
Johannes Berg	580	1.1%
Peter Zijlstra	554	1.1%
Bartlomiej Zolnierkiewicz	504	1.0%
Greg Kroah-Hartman	491	0.9%
Mark Brown	489	0.9%

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

Name	Number of Changes	Percent of Changes
Takahashi Iwai	454	0.9%
Mauro Carvalho Chehab	432	0.8%
Joe Perches	415	0.8%
Ingo Molnar	412	0.8%
Arnaldo Carvalho de Melo	392	0.8%
Roel Kluin	386	0.7%
Magus Damm	378	0.7%
Ben Dooks	364	0.7%
Alan Cox	363	0.7%
Frederic Weisbecker	361	0.7%
Tejun Heo	360	0.7%
Luis R. Rodriguez	358	0.7%
Eric Dumazet	347	0.7%
Julia Lawall	326	0.6%
Sage Weil	323	0.6%
David S. Miller	315	0.6%
Christoph Hellwig	313	0.6%
Alex Deucher	304	0.6%
Mike Frysinger	303	0.6%
Steven Rostedt	303	0.6%
Thomas Gleixner	291	0.5%
Dan Carpenter	273	0.5%
Ben Hutchings	272	0.5%
Randy Dunlap	256	0.5%

It is amusing to note that Linus Torvalds (886 total changes, 168 since 2.6.30) does not appear in the top-30 list. Linus remains an active and crucial part of the development process; his contribution cannot be measured just by the number of changes made. We are seeing a similar pattern with a number of other senior kernel developers; as they put more time into the review and management of patches from others, they write fewer patches of their own. (Obscure technical detail: these numbers do not count "merge commits," where one set of changes is merged into another. Linus Torvalds generates large numbers of merge commits; had these been counted he would have shown up on this list.)

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 the table above, an increasing number of companies are working toward the improvement of the kernel.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

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: (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 the table below. With few exceptions, all of the people in this category have contributed ten 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 nearly 70% 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.

Company Name	Number of Changes	Percent of Total
None	35,663	18.9%
Red Hat	23,356	12.4%
Novell	13,120	7.0%
IBM	13,026	6.9%
Unknown	12,060	6.4%
Intel	11,028	5.8%
consultants	4,817	2.6%
Oracle	4,367	2.3%
Renesas Technology	2,621	1.4%
The Linux Foundation	2,488	1.3%
academics	2,464	1.3%
SGI	2,450	1.3%
Fujitsu	2,293	1.2%
Parallels	2,226	1.2%
Analog Devices	1,955	1.0%
Nokia	1,896	1.0%
HP	1,854	1.0%
MontaVista	1,821	1.0%
Google	1,565	0.8%
AMD	1,518	0.8%
Freescale	1,501	0.8%

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

Company Name	Number of Changes	Percent of Total
linutronix	1,470	0.8%
MIPS Technologies	1,410	0.7%
NetApp	1,322	0.7%
Marvell	1,241	0.7%
Atheros Communications	1,234	0.7%
Astaro	1,222	0.6%
Broadcom	1,130	0.6%
QLogic	1,076	0.6%
NTT	1,068	0.6%

What we see here is that a small number of companies is responsible for a large portion of the total changes to the kernel. But there is a "long tail" of companies (over 500 of which do not appear in the above list) 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.

The picture since 2.6.30 shows some interesting changes:

Company Name	Number of Changes	Percent of Total
None	9,911	19.1%
Red Hat	6,219	12.0%
Intel	4,037	7.8%
Novell	2,625	5.0%
IBM	2,491	4.8%
unknown	2,456	4.7%
consultants	1,265	2.4%
Nokia	1,173	2.3%
Renesas Technology	1,032	2.0%
Oracle	995	1.9%
Fujitsu	904	1.7%
AMD	860	1.7%
Texas Instruments	775	1.5%
academics	774	1.4%
Atheros Communications	728	1.4%
Analog Devices	698	1.3%
HP	523	1.0%
Pengutronix	516	1.0%
Wolfson Microelectronics	488	0.9%
Broadcom	407	0.8%
NTT	406	0.8%

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

Company Name	Number of Changes	Percent of Total
Marvell	390	0.8%
NetApp	363	0.7%
New Dream Network	357	0.7%
MontaVista	349	0.7%
Google	340	0.7%
Samsung	335	0.6%
QLogic	335	0.6%
Societe Francaise du Radiotelephone	333	0.6%
Parallels	319	0.6%

The companies at the top of the listing are almost the same, and Red Hat maintains its commanding lead here. But we see companies like Nokia, AMD, Texas Instruments, and Samsung working up to higher contribution levels as they increase their investment in Linux kernel development.

This rise in development of Linux sponsored by embedded/mobile companies and their suppliers reflects the increasing importance of Linux in those markets.

Who is Reviewing the Work

Patches do not normally pass directly into the mainline kernel; instead, they pass through one of over 100 subsystem trees. Each subsystem tree is dedicated to a specific part of the kernel (examples might be SCSI drivers, x86 architecture code, or networking) and is under the control of a specific maintainer. When a subsystem maintainer accepts a patch into a subsystem tree, he or she will attach a "Signed-off-by" line to it. This line is a statement that the patch can be legally incorporated into the kernel; the sequence of signoff lines can be used to establish the path by which each change got into the kernel.

An interesting (if approximate) view of kernel development can be had by looking at signoff lines, and, in particular, at signoff lines added by developers who are not the original authors of the patches in question. These additional signoffs are usually an indication of review by a subsystem maintainer. Analysis of signoff lines gives a picture of who admits code into the kernel - who the gatekeepers are. Since 2.6.30, the developers who added the most non-author signoff lines are:

Name	Signoff Lines	Percent of Total	Subsystem
David S. Miller	5,153	10.9%	Networking, IDE, Sparc
John W. Linville	3,611	7.6%	Wireless Networking
Greg Kroah-Hartman	3,556	7.5%	USB, staging, driver core
Andrew Morton	3,042	6.4%	Everything
Ingo Molnar	2,887	6.1%	x86 architecture
Mauro Carvalho Chehab	2,353	5.0%	Video for Linux (Media Devices)
James Bottomley	1,294	2.7%	SCSI

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

Name	Signoff Lines	Percent of Total	Subsystem
Dave Airlie	876	1.9%	DRM (Video Drivers)
Paul Mundt	654	1.4%	SuperH architecture
Len Brown	651	1.4%	ACPI
Takashi Iwai	647	1.4%	Sound
Russell King	616	1.3%	ARM architecture
Jeff Kirsher	607	1.3%	Intel network drivers
Linus Torvalds	600	1.3%	Everything
Avi Kivity	573	1.2%	KVM
Benjamin Herrenschmidt	534	1.1%	PowerPC architecture
Mark Brown	526	1.1%	System-on-a-chip Sound
Reinette Chatre	524	1.1%	Intel wireless drivers
Ralf Baechle	502	1.1%	MIPS architecture
Peter Anvin	495	1.0%	x86 architecture
Jens Axboe	487	1.0%	Block layer
Jesse Barnes	454	1.0%	PCI
Mike Frysinger	441	0.9%	Blackfin architecture
Tony Lindgren	432	0.9%	OMAP architecture
Eric Anholt	425	0.9%	Intel video drivers
Ben Dooks	412	0.9%	Samsung/ARM architecture
David Woodhouse	406	0.9%	JFFS2, Embedded, MTD, IOMMU
Martin Schwidefsky	387	0.8%	S390 architecture
Kevin Hilman	360	0.8%	TI Davinci architecture
Jiri Kosina	332	0.7%	HID, Trivial

From this table, we see that Linus Torvalds directly merges just over 1% of the total patch stream; everything else comes in by way of the subsystem maintainers.

Associating signoffs with employers yields the following:

Company Name	Signoff Lines	Percent of Total
Red Hat	17,815	37.7%
Novell	6,345	13.4%
Intel	4,365	9,2%
Google	3,133	6.6%
none	2,268	4.8%
IBM	1,663	3.5%
consultant	1,153	2.4%
Oracle	870	1.8%
Renesas Technology	674	1.4%
The Linux Foundation	600	1.3%
Analog Devices	541	1.1%
Nokia	540	1.1%

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

Company Name	Signoff Lines	Percent of Total
Wind River	539	1.1%
Wolfson Microelectronics	526	1.1%
Atomide	432	0.9%
Simtec	408	0.9%
Marvell	406	0.9%
NetApp	351	0.7%
unknown	247	0.5%
Cisco	245	0.5%

The signoff metric is a loose indication of review, so the above numbers need to be regarded as approximations only. Still, one can clearly see that subsystem maintainers are rather more concentrated than kernel developers as a whole; over half of the patches going into the kernel pass through the hands of developers employed by just two companies.

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, Fujitsu, 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.

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.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

THE LINUX FOUNDATION

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 in response to number of different environments it is used in. This rate of change continues to increase, as does the number of developers and companies involved in the process; thus far, the development process has proved that it is able to scale up to higher speeds without trouble.

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 over time. The kernel development community welcomes new developers; individuals or corporations interested in contributing to the Linux kernel are encouraged to consult "How to participate in the Linux community" (which can be found at http://ldn.linuxfoundation.org/book/how-participate-linux-community) or to contact the authors of this paper or the Linux Foundation for more information.

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

Many of the statistics in this article were generated by the "gitdm" tool, written by Jonathan Corbet. Gitdm is distributable under the GNU GPL; it can be obtained from git://git.lwn.net/gitdm.git.

The information for this paper was retrieved directly from the Linux kernel releases as found at the http://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/

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org
compliance@linuxfoundation.org

The Linux Foundation promotes, protects and advances Linux by providing unified resources and services needed for open source to successfully compete with closed platforms.

To learn more about The Linux Foundation, and our other initiatives please visit us at http://www.linuxfoundation.org/.