Linux Kernel Development

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

Jonathan Corbet, LWN.net
Greg Kroah-Hartman, The Linux Foundation
Amanda McPherson, The Linux Foundation

March 2012

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 between 8,000 and 12,000 patches going into each recent kernel release. These releases each contain the work of over 1,000 developers representing nearly 200 corporations.

Since 2005, over 7,800 individual developers from almost 800 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.

This is the fourth update of this document, which has been published roughly annually since 2008. It covers development through the 3.2 release, with an emphasis on the releases (2.6.36 to 3.2) made since the last update. It has been a busy period, with seven kernel releases created, many significant changes made, and continual growth of the kernel developer and user community.

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.

Some 2011 Kernel Development Highlights

It was another busy year in the kernel development community, as will be seen in the statistics shown below. Beyond the sheer volume of changes made, though, there are a few other noteworthy things that happened during this time. They include:

- The Linux kernel celebrated its twentieth anniversary; the initial announcement from Linus Torvalds was posted on August 25, 1991. It was quickly followed by the first release (version 0.01) in September.
- The 2.6.x numbering scheme, which had been used since 2004, was finally put to rest with the release of the 3.0 kernel. There was nothing special about the 3.0 release; it was simply decided (after years of occasional discussion) that the 2.6 version numbers were getting unwieldy.
- For the first time ever, Microsoft appeared in the list of the top-20 contributors for a kernel release.
- The central repository and distribution site for kernel development - kernel.org - suffered a severe security breach and was offline for several weeks. As a result, the 3.1 kernel release was delayed. Extensive investigations have concluded that no attempt was made to compromise the integrity of the kernel source (such attempts would almost certainly have been discovered at the time anyway). Even so, as a result of this incident, the security of the development process has been strengthened in a number of ways.
- There was a well-publicized blow-up over the state of the ARM architecture subtree in the kernel. It is true that ARM has gotten a bit messy as the result of increased contributions from the embedded community. In a sense, the kernel is a victim of its own success. By the end of the year, the effort to clean up this code was already well advanced.
- The 2.0 release, in June, 1996, added symmetric multiprocessing support and, with it, the dreaded big kernel lock. In 2011, almost exactly fifteen years later, the process of removing that lock was completed with the release of the 2.6.39 kernel.
- The “Long-Term Support Initiative” was announced at the end of 2011. This effort, described in more detail below, will provide predictable support for some kernel releases with an emphasis on providing a stable base for embedded products.

Above and beyond all of that, though, the process of developing the kernel and making it better continued at a fast pace. The remainder of this document will concern itself with the health of the development process and where all that code came from.

Development Model

Linux kernel development proceeds under a loose, time-based release model, with a new major kernel release occuring 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 up 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.

After each major kernel release by Linus Torvalds, the kernel's "stable team" (currently Greg Kroah-Hartman) takes up short-term maintenance, applying important fixes as they are developed. The stable process 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 two development cycles and, for specific kernel releases, can go significantly longer. In recent years we have seen an increasing number of cooperative industry efforts to maintain specific kernels for periods of one year or more.

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
2.6.36	2010-10-20	80
2.6.37	2011-01-04	76
2.6.38	2011-03-14	69
2.6.39	2011-05-18	65
3.0	2011-07-21	64
3.1	2011-10-24	95
3.2	2012-01-04	72

The average kernel development cycle runs for about 80 days, just under twelve weeks. The length of the cycle has been slowly declining in recent years; many cycles now take less than 70 days to complete.

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
2.6.36	9,501
2.6.37	11,446
2.6.38	9,577
2.6.39	10,269
3.0	9,153
3.1	8,693
3.2	11,881

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
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
2.6.36	4.95
2.6.37	6.28
2.6.38	5.78
2.6.39	6.58
3.0	5.96
3.1	3.81
3.2	6.88

So, between the 2.6.11 and 3.2 kernel releases (which were 2,581 days apart), there were, on average, 4.3 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: 5.64 patches per hour. As the Linux kernel grows, the rate of change is growing with it.

It is 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 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	47
2.6.13	5	39
2.6.14	7	89
2.6.15	7	103
2.6.16	62	991
2.6.17	14	177
2.6.18	8	232
2.6.19	7	185
2.6.20	21	447
2.6.21	7	155
2.6.22	19	366
2.6.23	17	302
2.6.24	7	243
2.6.25	20	481
2.6.26	8	321
2.6.27	61	1,879
2.6.28	10	611
2.6.29	6	379
2.6.30	10	431
2.6.31	14	819
2.6.32	57	3,315
2.6.33	20	1,877
2.6.34	10	1,323
2.6.35	14	1,609
2.6.36	4	687
2.6.37	6	592
2.6.38	8	634
2.6.39	4	441
3.0	22	1,375
3.1	10	694
3.2	7	389

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 released kernels. Additionally, some kernels are receiving stable updates for relatively long periods of time. For example, 2.6.32 is still supported by a number of distributors, so it continues to receive updates.

A number of changes were made to the management of stable updates in 2011. Most kernel releases will now receive updates for two cycles, after which they will not be supported. Occasional kernels will be selected for longer-term maintenance, though.

The embedded community has developed a plan to select one kernel each year and maintain it for two years thereafter; the 3.0 release will be the first to be maintained in this manner. See **'The embedded long term support initiative'** for more information on this effort.

In summary, the stable update series continues to prove its value by allowing the final fixes to be made to released kernels while simultaneously letting mainline development move forward.

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
2.6.36	34,317	13,422,037
2.6.37	36,189	13,919,579
2.6.38	36,868	14,211,814
2.6.39	36,713	14,537,764
3.0	36,788	14,651,135
3.1	37,095	14,776,002
3.2	37,626	15,004,006

Since the first version of this paper, the kernel has grown by over 8 million lines of code - 1.5 million since the 2010 update. The kernel has, in fact, grown steadily since its first release - a mere 10,000 lines of code - came out in 1991. The one exception is 2.6.36, which is the only kernel release ever that was smaller than its predecessor. The reduction in size was the result of the clean up of a lot of default configuration files. Despite an ongoing effort to eliminate duplicated code and generally clean up the kernel tree, we are unlikely to see the source base shrink again anytime soon.

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.

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	249
2.6.31	1,166	227
2.6.32	1,248	261
2.6.33	1,196	238
2.6.34	1,150	243
2.6.35	1,187	209
2.6.36	1,176	207
2.6.37	1,276	221
2.6.38	1,198	220
2.6.39	1,258	239
3.0	1,131	331
3.1	1,168	212
3.2	1,316	226
All	7,944	855

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 9% of the total changes and the top 30 developers have contributed just over 20% 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:

Name	Number of Changes	Percent of Changes
David S. Miller	3,258	1.2%
Al Viro	2,840	1.1%
Takashi Iwai	2,637	1.0%
Ingo Molnar	2,348	0.9%
Tejun Heo	2,235	0.9%

Name	Number of Changes	Percent of Changes
Thomas Gleixner	2,190	0.8%
Paul Mundt	2,093	0.8%
Russell King	2,083	0.8%
Bartlomiej Zolnierkiewicz	2,074	0.8%
Adrian Bunk	2,042	0.8%
Stephen Hemminger	1,918	0.7%
Johannes Berg	1,915	0.7%
Greg Kroah-Hartman	1,899	0.7%
Mauro Carvalho Chehab	1,879	0.7%
Mark Brown	1,781	0.7%
Ralf Baechle	1,735	0.7%
Christoph Hellwig	1,716	0.7%
Alan Cox	1,703	0.6%
Andrew Morton	1,638	0.6%
Randy Dunlap	1,546	0.6%
Jean Delvare	1,467	0.6%
Joe Perches	1,456	0.6%
Hans Verkuil	1,307	0.5%
Ben Dooks	1,299	0.5%
Trond Myklebust	1,277	0.5%
Patrick McHardy	1,253	0.5%
Eric Dumazet	1,251	0.5%
Peter Zijlstra	1,237	0.5%
Neil Brown	1,182	0.5%
Mike Frysinger	1,163	0.4%

The above numbers are drawn from the entire git repository history, starting with 2.6.12. If we look at the commits since the last version of this paper (2.6.35) through 3.2, the picture is similar but not identical:

Name	Number of Changes	Percent of Changes
Mark Brown	887	1.3%
Thomas Gleixner	798	1.1%
Joe Perches	683	1.0%
Chris Wilson	639	0.9%
David S. Miller	636	0.9%
Axel Lin	632	0.9%
Eric Dumazet	614	0.9%
K. Y. Srinivasan	599	0.8%
Johannes Berg	581	0.8%
Al Viro	575	0.8%
Tejun Heo	537	0.8%
Ben Skeggs	536	0.8%
Dan Carpenter	519	0.7%
Takashi Iwai	517	0.7%
Mauro Carvalho Chehab	499	0.7%
Russell King	494	0.7%
Christoph Hellwig	450	0.6%
Jonathan Cameron	439	0.6%
Alex Deucher	422	0.6%
Larry Finger	391	0.6%
Felix Fietkau	382	0.6%
Roland Vossen	377	0.5%
Uwe Kleine-König	356	0.5%

Name	Number of Changes	Percent of Changes
Arend van Spriel	355	0.5%
Wey-Yi Guy	352	0.5%
Greg Kroah-Hartman	340	0.5%
Guennadi Liakhovetski	336	0.5%
Vasiliy Kulikov	325	0.5%
Randy Dunlap	324	0.5%
Hans Verkuil	308	0.4%

It is amusing to note that Linus Torvalds (1,113 total changes, 231 since 2.6.35) does not appear in either 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 these lists.)

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

Below we look more closely at the companies that 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" 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 60% 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 75% 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	46,982	17.9%
Red Hat	31,261	11.9%
Novell	16,738	6.4%
Intel	16,219	6.2%
IBM	16,073	6.1%
Unknown	13,342	5.1%
Consultant	7,986	3.0%
Oracle	5,542	2.1%
Academia	3,421	1.3%
Nokia	3,272	1.2%

Company Name	Number of Changes	Percent of Total
Fujitsu	3,156	1.2%
Texas Instruments	2,982	1.1%
Broadcom	2,916	1.1%
Linux Foundation	2,890	1.1%
Google	2,620	1.0%
Analog Devices	2,595	1.0%
SGI	2,578	1.0%
AMD	2,510	1.0%
Parallels	2,419	0.9%
Freescale	2,265	0.9%
Cisco	2,259	0.9%
HP	2,158	0.8%
Renesas Technology	2,092	0.8%
MontaVista	2,019	0.8%
Atheros Communications	1,960	0.7%
Wolfson Microelectronics	1,952	0.7%
Marvell	1,752	0.7%
NetApp	1,746	0.7%
Linutronix	1,656	0.6%
Samsung	1,650	0.6%

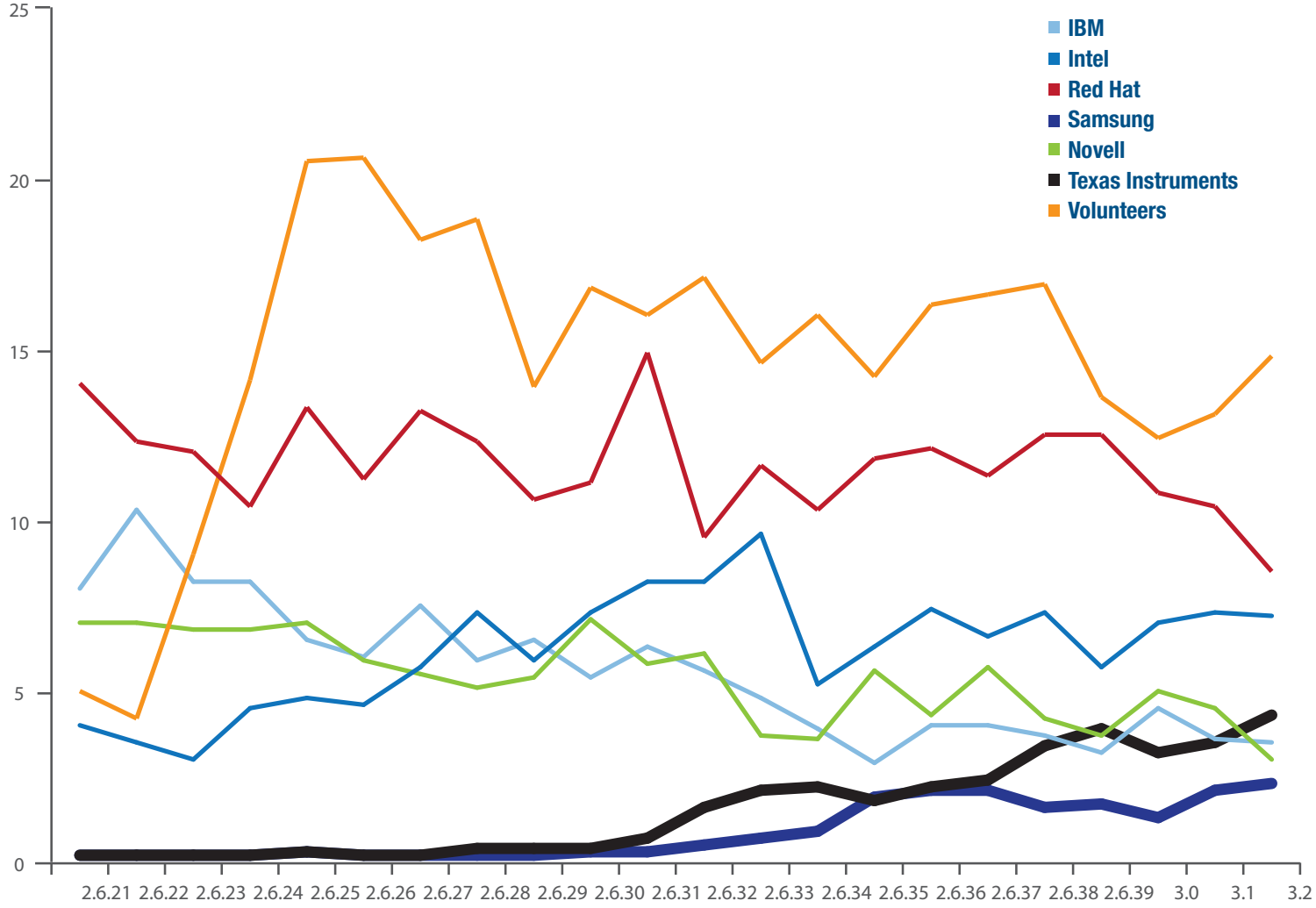
What we see here is that a small number of companies are responsible for a large portion of the total changes to the kernel. But there is a "long tail" of companies (over 700 of which do not appear in the above list) that have made significant changes. There may be no other examples of such a large, common resource being supported by such a large group of independent actors in such a collaborative way.

The picture since 2.6.36 shows some interesting changes:

Company Name	Number of Changes	Percent of Total
None	11,413	16.2%
Red Hat	7,563	10.7%
Intel	5,075	7.2%
Novell	3,050	3.3%
Unknown	2,998	4.3%
IBM	2,638	3.7%
Texas Instruments	2,124	3.0%
Consultant	1,859	2.6%
Broadcom	1,780	2.5%
Nokia	1,367	1.9%
Samsung	1,195	1.7%
Oracle	1,102	1.6%
Google	1,054	1.5%
Wolfson Microelectronics	1,005	1.4%
AMD	980	1.4%
Academia	882	1.3%
Fujitsu	854	1.2%
Pengutronix	733	1.0%

Company Name	Number of Changes	Percent of Total
Atheros Communications	726	1.0%
Freescale	712	1.0%
Microsoft	688	1.0%
ST Ericsson	663	0.9%
Wind River	645	0.9%
MiTAC	632	0.9%
Soc. Francaise de Radiotelephone	614	0.9%
Analog Devices	611	0.9%
tglx PITA	591	0.8%
Linaro	527	0.7%
QLogic	526	0.7%
Marvell	465	0.7%

The companies at the top of the listing are almost the same, and Red Hat maintains its commanding lead here. However, there is an interesting trend to be seen in the following plot:



This plot shows the percentage of changesets contributed by a number of the most active companies since the 2.6.20 release in 2007. The level of contribution is approximately equal for most of these companies over this time period - meaning that, as the pace of kernel development has increased, they have increased their contributions accordingly.

The highlighted traces at the bottom, though, show a different trend. They correspond to the contributions from Samsung and Texas Instruments, both of which are prominent mobile and embedded companies. In recent years, the level of participation from this sector has been growing rapidly. It is worth noting that these companies are not only adding more hardware support to the kernel, they are also taking more responsibility for the advancement of core kernel areas like the scheduler and memory management.

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 2.6.35, the developers who added the most non-author signoff lines are:

Name	Signoff Lines	Percent of Total
Greg Kroah-Hartman	7,848	5.8%
David S. Miller	6,246	4.6%
John W. Linville	4,146	3.1%
Linus Torvalds	3,266	2.4%
Mauro Carvalho Chehab	3,253	2.4%
Andrew Morton	2,687	2.0%
Mark Brown	2,131	1.6%
James Bottomley	1,609	1.2%
Takashi Iwai	1,282	0.9%
Russell King	1,245	0.9%
Ingo Molnar	1,216	0.9%
Thomas Gleixner	1,088	0.8%
Paul Mundt	1,029	0.8%
Dave Airlie	1,003	0.7%
Chris Wilson	944	0.7%
Al Viro	915	0.7%
Kukjun Kim	818	0.6%
Wey-Yi Guy	787	0.6%
Avi Kivity	762	0.6%
Artem Bityutskiy	761	0.6%

From this table, we see that Linus Torvalds directly merges just over 2% 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	26,225	37.7%
Novell	13,722	13.4%
None	13,587	9.2%

Company Name	Signoff Lines	Percent of Total
Intel	9,876	6.6%
IBM	4,801	4.8%
Google	4,057	3.5%
Texas Instruments	3,744	2.4%
Linux Foundation	3,270	1.8%
Unknown	3,161	1.4%
Consultant	2,899	1.3%
Samsung	2,463	1.1%
Wolfson Microelectronics	2,250	1.1%
Broadcom	2,198	1.1%
Microsoft	2,174	1.1%
Nokia	2,108	0.9%
Oracle	1,817	0.9%
Pengutronix	1,420	0.9%
Wind River	1,285	0.7%
Academia	1,126	0.5%
AMD	1,101	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 three companies.

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 the 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 that 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 that this number of contributing companies – and individual developers – will continue to increase. 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 Kernel 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, because 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 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 **https://github.com/gregkh/kernel-history**.

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/.