Linux Kernel Development

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

September 2013

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

This is the fifth update of this document, which has been published roughly annually since 2008. It covers development through the 3.10 release, with an emphasis on the releases (3.3 to 3.10) made since the last update.

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 over 10,000 patches going into each recent kernel release. These releases each contain the work of over 1100 developers representing over 225 corporations.

Since 2005, nearly 10,000 individual developers from over 1000 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 fifth update of this document, which has been published roughly annually since 2008. It covers development through the 3.10 release, with an emphasis on the releases (3.3 to 3.10) made since the last update. It has been a busy period, with eight 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 2012-13 Kernel Development Highlights

The kernel development community remains extremely busy, as will be seen in the statistics shown below. Some of the highlights worth noting since the last release of this paper (April 2012) include:

- Almost 92,000 changesets have been merged from 3,738 individual developers representing 536 corporations (that we know about).
- A vast array of important new features has been merged into the mainline. These include full tickless operation, user namespaces, KVM and Xen virtualization for ARM, per-entity load tracking in the scheduler, user-space checkpoint/restart, 64-bit ARM architecture support, the F2FS flash-oriented filesystem, many networking improvements aimed at the latency and bufferbloat problems, two independent subsystems providing fast caching for block storage devices, and much more.
- The longstanding squabble over Android-specific kernel features has faded completely into the background. The much-discussed "wakelocks" feature has been quietly replaced by a different mainline solution which is used in the latest Android devices.
- The use of automated tools to find bugs in development kernels has increased significantly during this period. Tools like the "trinity" fuzz tester and the zero-day build-and-boot system are finding large numbers of bugs in pre-release kernels, shortening the development cycle and enabling the community to deliver higher-quality releases.
- Contributions from the mobile and embedded industries continue to increase. Linaro, Samsung, and TI, for example, together contributed 4.4% of the changes in the previous version of this paper; for the period up to 3.10, they contributed almost 11% of all changes.
- The kernel project participated in the Outreach Program for Women for the first time, leading to 41 applications for 7 available positions. During the application process, 374 patches were submitted to the kernel, and over 1/3 of those patches were accepted in the 3.10 kernel release. The intern process is now underway, but the results of that will not start showing up until future kernel releases.

Above and beyond all of that, 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 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 mainline kernel 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 ensures that important fixes are made available to distributors and users and that they are incorporated into future mainline releases as well. 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 time 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, but that variation has decreased in recent years.

The release history for recent kernels is:

Kernel Version	Release Date	Days of Development
3.0	2011-07-21	64
3.1	2011-10-24	95
3.2	2011-12-31	68
3.3	2012-03-18	74
3.4	2012-05-20	63
3.5	2012-07-21	62
3.6	2012-09-30	71
3.7	2012-12-10	71
3.8	2013-02-18	70
3.9	2013-04-28	69
3.10	2013-06-30	63

Over time, kernel development cycles have slowly been getting shorter. The previous version of this report stated that the average cycle lasted about 80 days; now the average is much closer to 70 days. The only recent development cycle to last more than 74 days was 3.1, which was delayed in the aftermath of the kernel.org compromise in 2011. This trend is almost certainly the result of improved discipline both before and during the development cycle: higher-quality patches are being merged, and the community is doing a better job of fixing regressions quickly. The increased use of automatic testing tools is also helping the community to find (and address) problems more quickly.

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)
3.0	9,153
3.1	8,693
3.2	11,780
3.3	10,550
3.4	10,889
3.5	10,957
3.6	10,247
3.7	11,990
3.8	12,394
3.9	11,910
3.10	13,367

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
3.0	5.96
3.1	3.81
3.2	6.88
3.3	5.94
3.4	7.20
3.5	7.36
3.6	6.01
3.7	7.04
3.8	7.38
3.9	7.19
3.10	9.02

The overall rate for the period covered in the previous version of this paper (2.6.35 to 3.2) was 6.71 patches per hour. As can be seen from the tables above, the number of changes being merged into each release is growing over time, even as the development cycle is getting shorter, so, as one would expect, the number of changes per hour is growing. Since the release of the 3.2 kernel, the development community has been merging patches at an average rate of 7.14 per hour, though, as can be seen, the rate for the 3.10 cycle was significantly higher than that.

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 recent stable kernel update history looks like this:

Kernel Version	Updates	Fixes
3.0	94	3,764
3.1	10	694
3.2	50	3,943
3.3	8	699
3.4	60	3,122
3.5	7	824
3.6	11	762
3.7	10	724
3.8	13	1,000
3.9	11	751
3.10	10	670

The number of fixes going into the stable releases is high and getting higher as the discipline for routing fixes to the stable trees improves.

The normal policy for stable releases is that each kernel will receive stable updates for a minimum of one development cycle (actually, until the -rc1 release of the second cycle following the initial release); thus we see nine or ten approximately weekly updates for most kernel releases. Roughly once each year, one release is chosen to receive updates for an extended, two-year period; as of this writing, the 3.0, 3.4, and 3.10 kernels are being maintained in this manner, and the 3.0 kernel is likely to be retired this October.

It is worth noting that some other kernel releases have been adopted for stable maintenance outside of the normal stable process; in particular, the 3.2 kernel is currently being maintained by Ben Hutchings.

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
3.0	36,788	14,651,135
3.1	37,095	14,776,002
3.2	37,626	15,004,006
3.3	38,091	15,171,607
3.4	38,573	15,389,393
3.5	39,101	15,601,911
3.6	39,738	15,873,569
3.7	40,912	16,197,233
3.8	41,532	16,422,416
3.9	42,435	16,692,421
3.10	43,029	16,961,031

The kernel has grown steadily since its first release in 1991, when there were only about 10,000 lines of code. At almost 17 million lines, the kernel is almost two million lines larger than it was at the time of the previous version of this paper.

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.

Kernel Version	Developers	Companies
3.0	1,131	191
3.1	1,168	189
3.2	1,316	231
3.3	1,247	233
3.4	1,286	245
3.5	1,195	242
3.6	1,224	298
3.7	1,280	228
3.8	1,258	241
3.9	1,388	263
3.10	1,392	243

These numbers show a continuation of the steady increase in the number of developers contributing to each kernel release. Indeed, the 3.10 kernel saw the most developers ever, while 3.9 included the participation of the most companies ever.

Since the beginning of the git era (the 2.6.11 release in 2005), a total of 9,784 developers have contributed to the Linux kernel; those developers worked for a minimum of 1,064 companies.

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. Since the 2.6.11 release, the top ten developers have contributed 30,420 changes — 8.4% of the total. The top thirty developers contributed just over 18% of the total. Those developers are below, and the numbers are drawn from the entire git repository history, starting with 2.6.12

Name	Changes	Percentage
Al Viro	4,124	1.2%
David S. Miller	3,690	1.0%
Takashi Iwai	3,387	1.0%
Mark Brown	3,199	0.9%
Tejun Heo	2,888	0.8%
Johannes Berg	2,841	0.8%
Mauro Carvalho Chehab	2,718	0.8%
Russell King	2,526	0.7%
Greg Kroah-Hartman	2,502	0.7%
Thomas Gleixner	2,423	0.7%
H Hartley Sweeten	2,415	0.7%
Ingo Molnar	2,376	0.7%
Paul Mundt	2,268	0.6%
Bartlomiej Zolnierkiewicz	2,076	0.6%
Hans Verkuil	2,074	0.6%

Name	Changes	Percentage
Alan Cox	1,962	0.6%
Adrian Bunk	1,919	0.5%
Christoph Hellwig	1,877	0.5%
Ralf Baechle	1,840	0.5%
Joe Perches	1,839	0.5%
Eric Dumazet	1,771	0.5%
Axel Lin	1,731	0.5%
Andrew Morton	1,669	0.5%
Trond Myklebust	1,665	0.5%
Randy Dunlap	1,645	0.5%
Jean Delvare	1,539	0.4%
Arnaldo Carvalho de Melo	1,409	0.4%
Dan Carpenter	1,393	0.4%
Peter Zijlstra	1,359	0.4%
Andi Kleen	1,359	0.4%

If we look at the commits since the last version of this paper (3.2 through 3.10), the picture is somewhat different:

Name	Changes	Percentage
H Hartley Sweeten	2,107	2.3%
Mark Brown	1,418	1.5%
Al Viro	1,311	1.4%
Axel Lin	1,078	1.2%
Johannes Berg	926	1.0%
Mauro Carvalho Chehab	838	0.8%
Hans Verkuil	767	0.8%
Takashi Iwai	750	0.8%
Daniel Vetter	695	0.7%
Sachin Kamat	679	0.7%
Alex Elder	676	0.7%
Tejun Heo	660	0.7%
Greg Kroah-Hartman	655	0.6%
Dan Carpenter	568	0.6%
Laurent Pinchart	565	0.6%
Lars-Peter Clausen	561	0.6%
Jingoo Han	555	0.6%
Ben Skeggs	532	0.6%
David S. Miller	521	0.6%
Eric Dumazet	521	0.6%
Stephen Warren	513	0.6%
Russell King	495	0.5%
Felipe Balbi	476	0.5%
Wei Youngjun	473	0.5%
Kuninori Morimoto	459	0.5%
Guenter Roeck	445	0.5%
Shawn Guo	443	0.5%
Trond Myklebust	438	0.5%
Eric W. Biederman	428	0.5%
Tomi Valkeinen	408	0.4%

Note that many senior kernel developers, Linus Torvalds included, do not show up on these lists. These developers spend much of their time getting other developers' patches into the kernel; this work includes reviewing changes and routing accepted patches toward the mainline.

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.

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.

Company	Changes	Total
None	12,550	13.6%
Red Hat	9,483	10.2%
Intel	8,108	8.8%
Texas Instruments	3,814	4.1%
Linaro	3,791	4.1%
SUSE	3,212	3.5%
Unknown	3,032	3.3%
IBM	2,858	3.1%
Samsung	2,415	2.6%
Google	2,255	2.4%
Vision Engraving Systems	2,107	2.3%
Consultants	1,529	1.7%
Wolfson Microelectronics	1,516	1.6%
Oracle	1,248	1.3%
Broadcom	1,205	1.3%

Company	Changes	Total
NVidia	1,192	1.3%
Freescale	1,127	1.2%
Ingics Technology	1,075	1.2%
Renesas Electronics	1,010	1.1%
Qualcomm	965	1.0%
Cisco	871	0.9%
The Linux Foundation	840	0.9%
Academics	831	0.9%
AMD	820	0.9%
Inktank Storage	709	0.8%
NetApp	707	0.8%
LINBIT	705	0.8%
Fujitsu	694	0.7%
Parallels	684	0.7%
ARM	664	0.7%

The top 10 contributors, including the groups “unknown” and “none,” make up over 55% 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 80% of all kernel development is demonstrably done by developers who are being paid for their work.

Interestingly, the volume of contributions from unpaid developers has been in slow decline for many years. What was 14.6% in the previous version of this paper is 13.6% now. There are many possible reasons for this decline, but, arguably, the most plausible of those is quite simple: kernel developers are in short supply, so anybody who demonstrates an ability to get code into the mainline tends not to have trouble finding job offers. Indeed, the bigger problem can be fending those offers off. As a result, volunteer developers tend not to stay that way for long.

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 since the 3.2 release. 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.

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

Name	Signoffs	Percent
Greg Kroah-Hartman	9,965	12.5%
David S. Miller	6,133	7.7%
Mauro Carvalho Chehab	3,847	4.8%
John W. Linville	3,541	4.4%
Andrew Morton	3,403	4.3%
Mark Brown	2,479	3.1%
James Bottomley	1,246	1.6%
Daniel Vetter	1,192	1.5%
Ingo Molnar	1.012	1.3%
Samuel Ortiz	984	1.2%

Name	Signoffs	Percent
Dave Airlie	940	1.2%
Kyungmin Park	892	1.1%
Rafael J. Wysocki	891	1.1%
Arnaldo Carvalho de Melo	868	1.1%
Felipe Balbi	856	1.1%
Jiri Kosina	803	1.0%
Kukjin Kim	793	1.0%
Linus Walleij	779	1.0%
Benjamin Herrenschmidt	770	1.0%
Jeff Kirshner	760	1.0%

The total number of patches signed off by Linus Torvalds (568, or 0.7% of the total) has fallen over the years. That reflects the increasing amount of delegation to subsystem maintainers who do the bulk of the patch review and merging.

Associating signoffs with employers yields the following:

Company	Signoffs	Percent
Red Hat	20,369	25.7%
The Linux Foundation	9.561	12.0%
Intel	7,244	9.1%
Google	4,605	5.8%
None	4,155	5.2%
SUSE	3,275	4.1%
Samsung	2,684	3.4%
Wolfson Microelectronics	2,474	3.1%
Texas Instruments	2,372	3.0%
IBM	2,245	2.8%

Company	Signoffs	Percent
Linaro	2,231	2.8%
Parallels	1,258	1.6%
LINBIT	930	1.2%
Fusion-io	815	1.0%
Broadcom	807	1.0%
Consultants	752	0.9%
OLPC	640	0.8%
Wind River	595	0.7%
Pengutronix	587	0.7%
Marvell	580	0.7%

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 four companies. That said, subsystem maintainers are less concentrated than they once were, and that trend appears to be continuing.

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, mobile and embedded markets, it's reasonable to expect 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 community" (http://www.linuxfoundation.org/content/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 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.