LINUX FOUNDATION

Linux Kernel Development

How Fast is it Going, Who is Doing It, What Are They Doing and Who is Sponsoring the Work

A Linux Foundation publication
February 2015

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

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 over 10,000 patches going into each recent kernel release. Each of these releases contains the work of over 1,400 developers representing over 200 corporations.

Since 2005, some 11,800 individual developers from nearly 1,200 different companies have contributed to the kernel. The Linux kernel, thus, has become a common resource developed on a massive scale by companies which are fierce competitors in other areas.

This is the sixth in a series of regular updates to this document, which has been published roughly annually since 2008. It covers development through the 3.18 release (which came out on December 7, 2014), with an emphasis on the releases (3.11 to 3.18) made since the last update. It has been a typically busy period, with eight kernel releases created, many significant changes made, and continual growth of the kernel developer and user communities.

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 2013-14 kernel development highlights

The kernel development community remains extremely busy, as well be seen in the following statistics.

- Just over 96,000 changesets have been merged from 4,169 individual developers representing 518 corporations (that we know about). The number of changesets (in other words, the rate of change of the kernel) and number of developers have both increased from the previous report, but the number of participating companies is down slightly.
- As usual, a wide array of new features has been merged during this time period. Some of the highlights include the O_TMPFILE option for the creation of temporary files, NFS 4.2 support, virtualization support on the ARM64 architecture with Xen and KVM, the “zswap” compressed swap cache, support for using GPU rendering engines independently of a graphical display, the multiqueue block layer for improved high-end disk I/O performance, the “nftables” firewall that will eventually replace iptables, the realtime earliest-deadline-first scheduler, a vast array of networking improvements, a major reworking of the control group subsystem, “file sealing” support for secure interprocess communication, the “overlayfs” union filesystem, hundreds of new drivers, thousands of fixes, and more.
- The kernel testing infrastructure continues to improve. The “zero-day build and boot robot” system alone found nearly 500 bugs (all of which were fixed) during this period. There is a rudimentary self-test framework in the kernel now that will be improved considerably in the coming year.

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.

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, most distributor kernels contain relatively few distribution-specific changes; this leads to higher quality and fewer differences between distributions.

After each mainline release, the kernel's "stable team" (currently led by 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 Release	Version Date	Days of development
3.11	2013-09-02	64
3.12	2013-11-03	62
3.13	2014-01-19	77
3.14	2014-03-30	70
3.15	2014-06-08	70
3.16	2014-08-03	56
3.17	2014-10-05	63
3.18	2014-12-07	63

Over time, kernel development cycles have slowly been getting shorter. The previous version of this report stated that the average cycle lasted about 70 days; now the average is just under 66 days.

One could argue that a number of kernels in 2014 might have been released even more quickly had the development cycle not aligned poorly with important developer conferences.

The trend toward shorter release cycles 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 large and increasing, as can be seen in the table below:

Kernel Version	Changes (patches)
3.11	10,893
3.12	10,927
3.13	12,127
3.14	12,311
3.15	13,722
3.16	12,804
3.17	12,354
3.18	11,379

The 3.15 development cycle was the busiest ever in the kernel’s history. 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
3.11	7.09
3.12	7.34
3.13	6.56
3.14	7.33
3.15	8.17

Kernel Version	Changes per Hour
3.16	9.53
3.17	8.17
3.18	7.53

The overall rate for the period covered in the previous version of this paper (3.2 to 3.10) was 7.14 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 3.10 kernel, the development community has been merging patches at an average rate of 7.71 patches per hour.

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 Release	Updates	Fixes
3.10	65	4,008
3.11	10	688
3.12	36	3,969
3.13	11	908
3.14	29	2,563
3.15	10	701
3.16	7	876
3.17	8	890
3.18	3	252

The number of updates for the 3.18 release is low because that kernel was quite new when this report was written.

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 have roughly nine approximately weekly updates for most kernel releases. About once each year, one release is chosen to receive updates for an extended, two-year period; as of this writing, the 3.10 and 3.14 kernels are being maintained in this manner.

It is worth noting that some other kernel releases have been adopted for stable maintenance outside of the normal stable process. In the above list, the large number of updates for 3.12 results from its ongoing maintenance by Jiri Slaby.

In the end, most Linux users are running a kernel based off one of the stable updates; to do otherwise would be to miss out on large numbers of important fixes. 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 Release	Files	Lines
3.11	44,017	17,407,037
3.12	44,601	17,730,630
3.13	44,985	17,934,674
3.14	45,950	18,275,747
3.15	46,795	18,636,331
3.16	47,440	18,882,881
3.17	47,505	18,868,140
3.18	47,986	18,997,848

The kernel has grown steadily since its first release in 1991, when there were only about 10,000 lines of code. At almost 19 million lines (up from 17 million), the kernel is almost two million lines larger than it was at the time of the previous version of this paper.

Sharp-eyed readers may note that the number of lines of code actually fell slightly in the 3.17 release; that was the result of the removal of a number of old and unmaintained drivers. That is only the second time in the entire history of kernel development that the kernel has gotten smaller; the first was the release of 2.6.36 in 2010.

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 Release	Developers	Companies
3.11	1,266	225
3.12	1,332	244
3.13	1,361	228
3.14	1,446	240
3.15	1,492	237
3.16	1,477	234
3.17	1,433	241
3.18	1,458	239

These numbers show a continuation of the steady increase in the number of developers contributing to each kernel release—we have nearly 200 more developers participating in each development cycle at the end of the study period than the beginning.

Since the beginning of the Git era (the 2.6.11 release in 2005), a total of 11,695 developers have contributed to the Linux kernel; those developers worked for a minimum of 1,230 companies. Interestingly, the number of companies supporting work on the kernel appears to be declining slowly, suggesting that developers are consolidating under a (slightly) smaller number of employers.

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 36,664 changes — 8.2% of the total. The top thirty developers contributed just over 17% of the total.

Those developers are:

Name	Changes	Percent
H Hartley Sweeten	4,967	1.1%
Al Viro	4,767	1.0%
Takashi Iwai	4,105	0.9%
Mark Brown	3,866	0.8%
David S. Miller	3,849	0.8%
Tejun Heo	3,492	0.8%
Johannes Berg	3,299	0.7%
Mauro Carvalho Chehab	3,275	0.7%
Russell King	3,051	0.7%
Greg Kroah-Hartman	2,993	0.7%
Thomas Gleixner	2,752	0.6%
Hans Verkuil	2,667	0.6%

Name	Changes	Percent
Joe Perches	2,488	0.5%
Ingo Molnar	2,474	0.5%
Axel Lin	2,271	0.5%
Paul Mundt	2,268	0.5%
Bartlomiej Zolnierkiewicz	2,221	0.5%
Christoph Hellwig	2,206	0.5%
Eric Dumazet	2,129	0.5%
Sachin Kamat	2,065	0.4%
Dan Carpenter	1,991	0.4%
Ralf Baechle	1,990	0.4%
Trond Myklebust	1,965	0.4%
Laurent Pinchart	1,936	0.4%
Adrian Bunk	1,919	0.4%
Alex Deucher	1,880	0.4%
Jingoo Han	1,837	0.4%
Daniel Vetter	1,770	0.4%
Andrew Morton	1,750	0.4%
Randy Dunlap	1,716	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 (3.10) through 3.18, the picture is somewhat different:

Name	Changes	Percent
H Hartley Sweeten	2,089	2.2%
Sachin Kamat	1,374	1.4%
Jingoo Han	1,230	1.3%
Laurent Pinchart	953	1.0%
Jes Sorensen	772	0.8%
Daniel Vetter	764	0.8%
Malcolm Priestley	745	0.8%
Alex Deucher	727	0.8%
Lars-Peter Clausen	697	0.7%
Geert Uytterhoeven	685	0.7%
Ville Syrjälä	669	0.7%
Mark Brown	653	0.7%
Takashi Iwai	601	0.6%
Tejun Heo	594	0.6%
Joe Perches	581	0.6%
Dan Carpenter	538	0.6%
Axel Lin	526	0.5%
Al Viro	524	0.5%
Russell King	517	0.5%

Name	Changes	Percent
Hans Verkuil	512	0.5%
Mauro Carvalho Chehab	511	0.5%
Fabio Estevam	507	0.5%
Johan Hedberg	502	0.5%
Navin Patidar	483	0.5%
Greg Kroah-Hartman	477	0.5%
Linus Walleij	473	0.5%
Ben Skeggs	458	0.5%
Fabian Frederick	457	0.5%
Marcel Holtmann	436	0.5%
Kuninori Morimoto	434	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 the following methods: (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	11,968	12.4%
Intel	10,108	10.5%
Red Hat	8,078	8.4%
Linaro	5,415	5.6%
Samsung	4,290	4.4%
Unknown	3,842	4.0%
IBM	3,081	3.2%
SUSE	2,890	3.0%
Consultants	2,451	2.5%
Texas Instruments	2,269	2.4%
Vision Engraving Systems	2,089	2.2%
Google	2,048	2.1%
Renesas Electronics	2,004	2.1%
Freescale	1,690	1.8%
Free Electrons	1,463	1.5%
FOSS Outreach Program for Women	1,418	1.5%
Oracle	1,166	1.2%
AMD	1,109	1.1%
NVidia	1,078	1.1%
Broadcom	1,001	1.0%
Huawei Technologies	971	1.0%
ARM	788	0.8%
Pengutronix	763	0.8%
Cisco	723	0.7%
Qualcomm	679	0.7%
Fujitsu	672	0.7%
Linux Foundation	627	0.6%
Imagination Technologies	579	0.6%
QLogic	545	0.6%
Ingics Technology	526	0.5%

The top 10 contributors, including the groups “unknown” and “none,” make up nearly 57% 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, well 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. It was 14.6% in the 2012 version of this paper, and 13.6% in 2013; now it is 11.8%.

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 400 of which do not appear in the above list) which have made significant changes since the 3.10 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.

Bringing in New Developers

The decline in volunteer developers mentioned in the previous section is potentially a cause for concern. Many, if not most of the current development community started that way, after all; might a shortage of volunteers lead to a shortage of kernel developers in the future?

The situation is worth watching, but there are a number of reasons to not worry too much about it at this time. The first of those was mentioned above: successful volunteers tend not to stay volunteers for long; why do the work for free when somebody is willing to pay for it? But there is more to the story than that.

Over the course of kernel development since the use of Git began, each kernel release has included contributions from 200–300 developers who had never put a patch into the kernel before. Outliers include 2.6.25 (333 new developers) and 2.6.20 (169 new developers). In the 3.x era, only 3.4 (with 182) has featured the work of less than 200 new developers.

For the time period covered by this paper, the history is:

Kernel Version	New developers
3.11	205
3.12	219
3.13	219
3.14	255
3.15	261
3.16	272
3.17	262
3.18	270

That adds up to 1,963 first-time developers over the course of about fifteen months. Remember that 4,171 developers overall contributed to the kernel during this time; one can thus conclude that nearly half of them were contributing for the first time.

Many of those developers will get their particular fix merged and never be seen again, but others will become permanent members of the kernel development community. Of those 1,963 new developers, 169 were known to be working on their own time, while we have not yet been able to get information on 778 of them. The rest of the new developers (1,016 — just over half) were already working for a company when they contributed their first patch to the kernel.

The companies that have been most active in bringing new developers into the community are:

Company	# New devs
Intel	147
Samsung	48
IBM	47
Google	43
Huawei Technologies	37
Red Hat	32
Freescale	31
Linaro	26
Texas Instruments	23
Marvell	15
NVIDIA	15

The FOSS Outreach Program for Women was responsible for introducing 24 new developers to the kernel community during this time. The sponsors of the program's kernel fellowships were Codethink (one), Intel (three) and Linux Foundation (three).

The bottom line is that even if all of the unknowns were volunteers, more than half of our new developers are paid to work on the kernel from their very first patch. In other words, companies working in this area have realized that one of the best ways to find new kernel development talent is to develop it in-house.

So, for many developers, employment comes first, and it is no longer necessary to put in time as a volunteer developer. This fact, too, can explain the decrease in volunteers over time while simultaneously showing that the community as a whole remains healthy.

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

Developer	Signoffs	Percent
Greg Kroah-Hartman	13,028	14.4%
David S. Miller	7,780	8.6%
Mark Brown	3,735	4.1%
Andrew Morton	3,726	4.1%
Mauro Carvalho Chehab	2,706	3.0%
Daniel Vetter	2,554	2.8%
John W. Linville	2,288	2.5%
Rafael J. Wysocki	1,614	1.8%
Simon Horman	1,339	1.5%
Ingo Molnar	1,243	1.4%
Linus Walleij	1,213	1.3%
Arnaldo Carvalho de Melo	1,044	1.2%
Jeff Kirsher	916	1.0%
Benjamin Herrenschmidt	906	1.0%
Shawn Guo	905	1.0%
Jonathan Cameron	871	1.0%
Felipe Balbi	861	1.0%
Jason Cooper	783	0.9%
Chris Mason	761	0.8%
Johannes Berg	748	0.8%

The total number of patches signed off by Linus Torvalds (329, or 0.4% of the total) continues its long-term decline. 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	16,963	18.8%
Linux Foundation	13,357	14.8%
Intel	11,045	12.2%
Linaro	8,422	9.3%
Google	5,207	5.8%
Samsung	4,728	5.2%
None	3,372	3.7%
SUSE	2,653	2.9%
IBM	2,208	2.4%
Texas Instruments	1,948	2.2%
Renesas Electronics	1,409	1.6%
Consultants	1,362	1.5%
Facebook	1,006	1.1%

Company	Signoffs	Percent
Broadcom	922	1.0%
University of Cambridge	871	1.0%
Unknown	805	0.9%
Parallels	734	0.8%
Fusion-IO	684	0.8%
Pure Storage	620	0.7%
Cisco	543	0.6%

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. Perhaps the most significant trend in this area is the increasing presence of the mobile and embedded sector.

Developers from these companies have been contributing changes at a high rate for some years now, but it has naturally taken longer for them to work up to the level of subsystem maintenance.

One could well argue that firms involved with enterprise computing still have a dominant role in the direction of kernel development, but the influence of mobile and embedded companies is on the rise.

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 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" (which can be found at **www.linuxfoundation.org/content/how-participate-linux-community**) or to contact the authors of this paper or The Linux Foundation for more information.

Authors

The report is co-authored by Jon Corbet, Linux kernel developer and editor of LWN.net; Greg Kroah-Hartman, Linux kernel maintainer and Linux Foundation fellow; and Amanda McPherson, chief marketing officer at The Linux Foundation.

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 website 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 **github.com/gregkh/kernel-history**

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

To learn more about The Linux Foundation or our other initiatives please visit us at www.linuxfoundation.org