
# Linux Kernel Development

## How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It: An August 2009 Update

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


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 1000 developers representing around 200 corporations.

Since 2005, over 5000 individual developers from nearly 500 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.

A number of changes have been noted since this paper was first published in 2008:

- We have seen a roughly 10% increase in the number of developers contributing to each kernel release cycle.
- The rate of change has increased significantly; the number of lines of code added to the kernel each day has nearly tripled.
- The kernel code base has grown by over 2.7 million lines

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

## 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 four years of kernel history as represented by the 2.6.11 through 2.6.30 releases. This is the second version of this paper, following up on **http://www.linuxfoundation.org/publications/linuxkerneldevelopment.php** which was published in April, 2008, and covered development through the 2.6.24 kernel. A look at the six kernel releases which have happened since then shows that, while many things remain the same, others are changing. In particular, the pace of development of the Linux kernel continues to increase.

## 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.

One significant change since the previous 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, 2.6.31 is in the stabilization phase, so linux-next contains changes intended for 2.6.32. 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 made up of Greg Kroah-Hartman and Chris Wright) 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 one development cycle and, for specific kernel releases, can go significantly longer.

This paper focuses exclusively on the mainline 2.6.x releases, to the exclusion of the stable updates. Those updates are small, and, in any case, the design of the development process requires that fixes accepted for -stable also be accepted into the mainline for the next major release.

## RELEASE FREQUENCY

The desired release period for a major kernel release is, by common consensus, 8–12 weeks. 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:

TABLE 1

| 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 |

| Kernel Version | Release Date | Days of Development |
|---|---|---|
| 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 |

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

## 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 Table 2.

TABLE 2

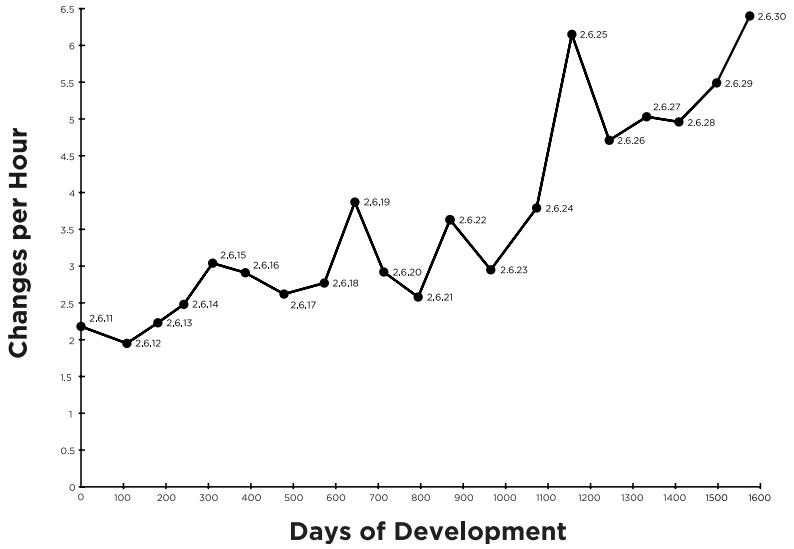
| Kernel Version | Changes (patches) |
|---|---|
| 2.6.11 | 3,616 |
| 2.6.12 | 5,047 |
| 2.6.13 | 3,904 |
| 2.6.14 | 3,627 |
| 2.6.15 | 4,959 |
| 2.6.16 | 5,369 |
| 2.6.17 | 5,727 |
| 2.6.18 | 6,323 |
| 2.6.19 | 6,685 |
| 2.6.20 | 4,768 |
| 2.6.21 | 5,016 |
| 2.6.22 | 6,526 |
| 2.6.23 | 6,662 |
| 2.6.24 | 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 |

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 Table 3:

TABLE 3

| 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 |

So, between the 2.6.11 and 2.6.30 kernel releases (which were 1560 days apart), there were, on average, 3.83 patches applied to the kernel tree per hour. In the time since the publication of the first version of this paper, that rate has been significantly higher: 5.45 patches per hour. As the Linux kernel grows, the rate of change is growing with it.




## KERNEL SOURCE SIZE

It is worth noting that this figure understates 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. 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.

TABLE 4

| 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 |

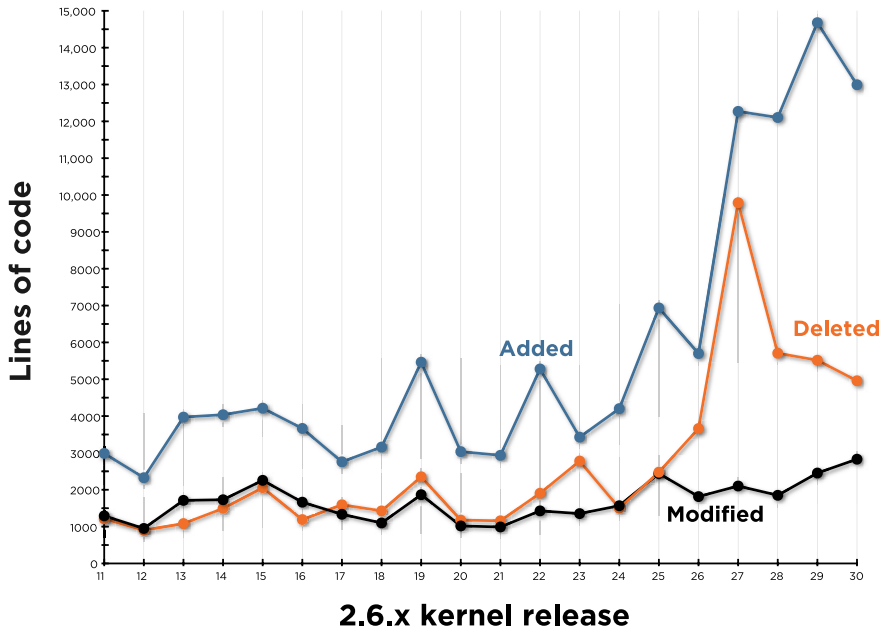
Since the first version of this paper, the kernel has grown by over 2.7 million lines of code. The growth rate of the kernel has always been high, but it increased significantly after the 2.6.27 kernel release. The main reason for this change is the addition of the staging tree, which has brought over 800,000 lines of previously out-of-tree code into the mainline kernel.

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 Table 5:

TABLE 5

| 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 |

Summing up these numbers, it comes to an impressive 6,422 lines added, 3,285 lines removed, and 1,687 lines changed every day for the past 4 1/2 years. Since 2.6.24, those number jump to an amazing 10,923 lines added, 5,547 lines removed, and 2,243 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 (the identification of the different companies is described in the next section) 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.

TABLE 6

| 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 | 194 |
| 2.6.25 | 1,123 | 228 |
| 2.6.26 | 1,027 | 203 |
| 2.6.27 | 1,021 | 188 |
| 2.6.28 | 1,075 | 213 |
| 2.6.29 | 1,180 | 230 |
| 2.6.30 | 1,150 | 240 |
| All | 4,910 | 532 |

These numbers show a consistent 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 4.5 years, the top 10 individual developers have contributed almost 12% of the number of changes and the top 30 developers have contributed over 25%. The list of individual developers, the number of changes they have contributed, and the percentage of the overall total can be seen in Table 7.

TABLE 7

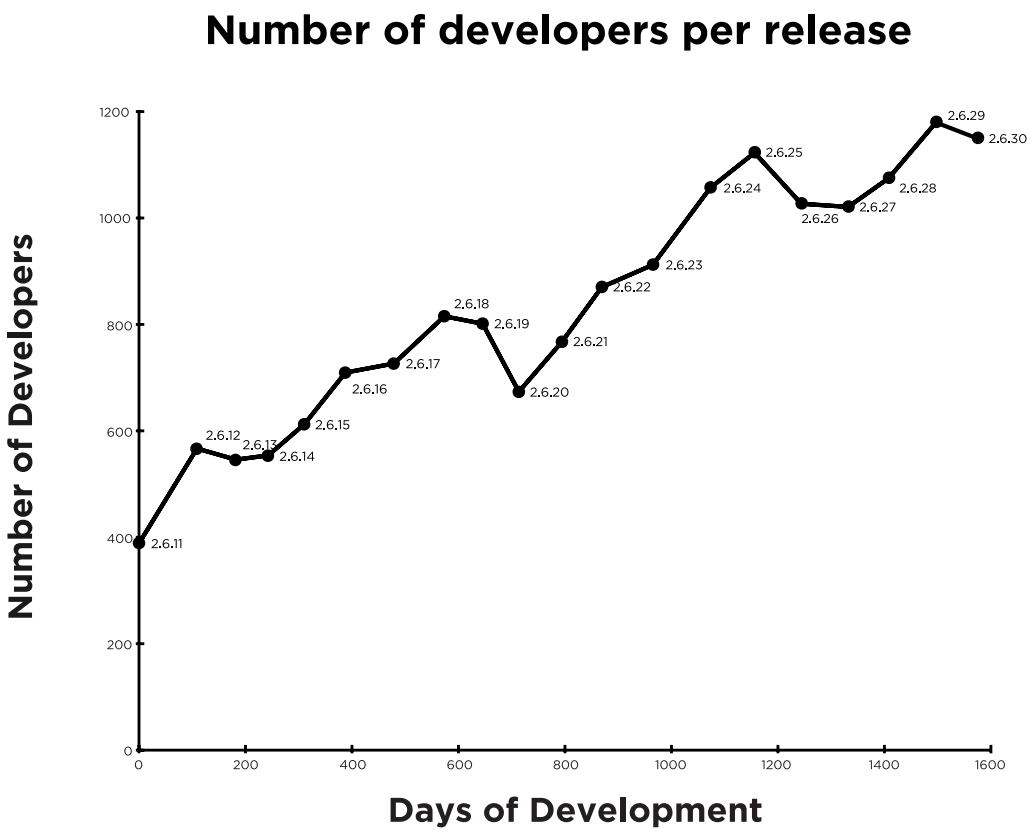
| Name | Number of Changes | Percent of Total Changes |
|---|---|---|
| David S. Miller | 2,239 | 1.5% |
| Ingo Molnar | 2,125 | 1.5% |
| Al Viro | 1,981 | 1.4% |
| Adrian Bunk | 1,883 | 1.3% |
| Takashi Iwai | 1,801 | 1.2% |
| | | |
| Bartlomiej Zolnierkiewicz | 1,651 | 1.1% |
| Ralf Baechle | 1,471 | 1.0% |
| Tejun Heo | 1,457 | 1.0% |
| Stephen Hemminger | 1,408 | 1.0% |
| Andrew Morton | 1,370 | 0.9% |
| | | |
| Paul Mundt | 1,331 | 0.9% |
| Russell King | 1,173 | 0.8% |
| Thomas Gleixner | 1,164 | 0.8% |
| Alan Cox | 1,145 | 0.8% |
| Greg Kroah-Hartman | 1,100 | 0.8% |
| | | |
| Patrick McHardy | 1,087 | 0.8% |
| Andi Kleen | 1,030 | 0.7% |
| Jean Delvare | 985 | 0.7% |
| Mauro Carvalho Chehab | 972 | 0.7% |
| Christoph Hellwig | 970 | 0.7% |
| | | |
| Randy Dunlap | 942 | 0.7% |
| Ben Dooks | 906 | 0.6% |
| David Woodhouse | 889 | 0.6% |
| Johannes Berg | 880 | 0.6% |
| David Brownell | 872 | 0.6% |
| | | |
| Hans Verkuil | 856 | 0.6% |
| Trond Myklebust | 844 | 0.6% |
| Michael Krufky | 814 | 0.6% |
| Alexey Dobriyan | 806 | 0.6% |
| Herbert Xu | 805 | 0.6% |

The above numbers are drawn from the entire git repository history, starting with 2.6.12. If we look at the commits since the first version of this paper (2.6.24) through 2.6.30, the picture is similar but not identical (Table 8):

TABLE 8

| Name | Number of Changes | Percent of Total Changes |
|---|---|---|
| Bartlomiej Zolnierkiewicz | 1,169 | (1.8%) |
| Ingo Molnar | 1,164 | (1.8%) |
| David S. Miller | 851 | (1.3%) |
| Chris Mason | 717 | (1.1%) |
| Takashi Iwai | 711 | (1.1%) |
| | | |
| Adrian Bunk | 708 | (1.1%) |
| Harvey Harrison | 669 | (1.0%) |
| Yinghai Lu | 609 | (0.9%) |
| Greg Kroah-Hartman | 608 | (0.9%) |
| Paul Mundt | 570 | (0.9%) |
| | | |
| Al Viro | 550 | (0.8%) |
| Alan Cox | 545 | (0.8%) |
| Stephen Hemminger | 522 | (0.8%) |
| Hans Verkuil | 517 | (0.8%) |
| Steven Rostedt | 511 | (0.8%) |
| | | |
| Mauro Carvalho Chehab | 503 | (0.8%) |
| Johannes Berg | 483 | (0.7%) |
| Jeremy Fitzhardinge | 465 | (0.7%) |
| Ben Dooks | 438 | (0.7%) |
| Tejun Heo | 430 | (0.7%) |
| | | |
| Pavel Emelyanov | 417 | (0.6%) |
| Mike Frysinger | 410 | (0.6%) |
| Glauber Costa | 406 | (0.6%) |
| Patrick McHardy | 404 | (0.6%) |
| Michael Krufky | 398 | (0.6%) |
| | | |
| Alexey Dobriyan | 391 | (0.6%) |
| Mark Brown | 390 | (0.6%) |
| Jean Delvare | 389 | (0.6%) |
| Thomas Gleixner | 388 | (0.6%) |
| Christoph Hellwig | 385 | (0.6%) |

It is amusing to note that Linus Torvalds (729 total changes, 254 since 2.6.24) fell off the top-30 list since the previous version of this report. Linus remains an active and crucial part of the development process; his contribution cannot be measured just by the number of changes made. (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.) Linus of course does a great deal of reviewing and sign offs of code. Please see the section “Who is Approving the Work” for further explanation of this.




## 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 on the next page. With few exceptions, all of the people in this category have contributed 10 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.

TABLE 9

| Company Name | Number of Changes | Percent of Total |
|---|---|---|
| None | 26,644 | 18.2% |
| Red Hat | 17,981 | 12.3% |
| Unknown | 11,164 | 7.6% |
| IBM | 11,151 | 7.6% |
| Novell | 11,046 | 7.6% |
| | | |
| Intel | 7,782 | 5.3% |
| Consultant | 3,657 | 2.5% |
| Oracle | 3,513 | 2.4% |
| Linux Foundation | 2,345 | 1.6% |
| SGI | 2,317 | 1.6% |
| | | |
| Parallels | 1,939 | 1.3% |
| Renesas Technology | 1,925 | 1.3% |
| Academia | 1,712 | 1.2% |
| Fujitsu | 1,592 | 1.1% |
| MontaVista | 1,564 | 1.1% |
| | | |
| MIPS Technologies | 1,537 | 1.1% |
| Analog Devices | 1,467 | 1.0% |
| HP | 1,415 | 1.0% |
| Freescale | 1,375 | 0.9% |
| Google | 1,261 | 0.9% |
| | | |
| linutronix | 1,246 | 0.9% |
| Astaro | 1,109 | 0.8% |
| NetApp | 1,049 | 0.7% |
| Marvell | 894 | 0.6% |
| Nokia | 842 | 0.6% |
| | | |
| Simtec | 820 | 0.6% |
| QLogic | 808 | 0.6% |
| Movial | 776 | 0.5% |
| AMD | 775 | 0.5% |
| Sun | 764 | 0.5% |

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 (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 2.6.24 shows some interesting changes (Table 10):

TABLE 10

| Company Name | Number of Changes | Percent of Total |
|---|---|---|
| None | 13,850 | 21.1% |
| Red Hat | 7,897 | 12.0% |
| IBM | 4,150 | 6.3% |
| Novell | 4,021 | 6.1% |
| Intel | 3,923 | 6.0% |
| | | |
| Unknown | 2,765 | 4.2% |
| Oracle | 2,003 | 3.1% |
| Consultant | 1,480 | 2.3% |
| Parallels | 1,142 | 1.7% |
| Fujitsu | 1,007 | 1.5% |
| | | |
| Academia | 992 | 1.5% |
| Analog Devices | 889 | 1.4% |
| Renesas Technology | 884 | 1.3% |
| SGI | 755 | 1.2% |
| Movial | 738 | 1.1% |
| | | |
| Sun | 639 | 1.0% |
| HP | 628 | 1.0% |
| Freescale | 613 | 0.9% |
| Marvell | 601 | 0.9% |
| MontaVista | 574 | 0.9% |
| | | |
| AMD | 552 | 0.8% |
| Nokia | 549 | 0.8% |
| Vyatta | 513 | 0.8% |
| Google | 512 | 0.8% |
| Atheros Communications | 494 | 0.8% |
| | | |
| NTT | 445 | 0.7% |
| linutronix | 445 | 0.7% |
| XenSource | 432 | 0.7% |
| Simtec | 414 | 0.6% |
| Astaro | 411 | 0.6% |

The increase in the number of developers with no employer is most likely an artifact of better information in our database—many of them were previously in the “unknown” category. 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 Oracle and Fujitsu working up to higher contribution levels and the welcome addition of companies like Atheros which have made the decision to support their products in the mainline Linux kernel.

## WHO IS REVIEWING THE WORK

Patches do not normally pass directly into the mainline kernel; instead, they pass through one of one-hundred or so 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 2.6.24, the developers who added the most non-author signoff lines are (Table 11):

TABLE 11

| | | |
|---|---|---|
| Andrew Morton | 6,515 | 10.5% |
| Ingo Molnar | 6,174 | 9.9% |
| David S. Miller | 5,954 | 9.6% |
| John W. Linville | 3,733 | 6.0% |
| Mauro Carvalho Chehab | 3,363 | 5.4% |
| | | |
| Greg Kroah-Hartman | 2,394 | 3.8% |
| Jeff Garzik | 1,879 | 3.0% |
| Thomas Gleixner | 1,707 | 2.7% |
| Linus Torvalds | 1,664 | 2.7% |
| James Bottomley | 1,302 | 2.1% |
| | | |
| Takashi Iwai | 1,093 | 1.8% |
| Len Brown | 1,033 | 1.7% |
| Russell King | 891 | 1.4% |
| Paul Mackerras | 876 | 1.4% |
| Bryan Wu | 806 | 1.3% |
| | | |
| Avi Kivity | 739 | 1.2% |
| Jaroslav Kysela | 660 | 1.1% |
| Martin Schwidefsky | 504 | 0.8% |
| Ralf Baechle | 495 | 0.8% |
| Bartlomiej Zolnierkiewicz | 486 | 0.8% |
| | | |
| Jesse Barnes | 477 | 0.8% |
| David Woodhouse | 474 | 0.8% |
| Roland Dreier | 472 | 0.8% |
| Paul Mundt | 467 | 0.8% |
| Jens Axboe | 402 | 0.6% |
| | | |

| | | |
|---|---|---|
| Patrick McHardy | 395 | 0.6% |
| Lachlan McIlroy | 383 | 0.6% |
| Kumar Gala | 381 | 0.6% |
| Benjamin Herrenschmidt | 378 | 0.6% |
| Mark Fasheh | 376 | 0.6% |

From this table, we see that Linus Torvalds directly merges just under 3% of the total patch stream; everything else comes in by way of the subsystem maintainers.

Associating signoffs with employers yields the following (Table 12):

TABLE 12

| | | |
|---|---|---|
| Red Hat | 22,652 | 36.4% |
| Google | 6,530 | 10.5% |
| Novell | 5,076 | 8.2% |
| None | 4,717 | 7.6% |
| Intel | 3,986 | 6.4% |
| | | |
| IBM | 3,321 | 5.3% |
| linutronix | 1,741 | 2.8% |
| Linux Foundation | 1,666 | 2.7% |
| Consultant | 1,213 | 1.9% |
| Hansen Partnership | 965 | 1.6% |
| | | |
| Analog Devices | 851 | 1.4% |
| SGI | 793 | 1.3% |
| Oracle | 725 | 1.2% |
| MIPS Technologies | 497 | 0.8% |
| Renesas Technology | 486 | 0.8% |
| | | |
| Qumranet | 477 | 0.8% |
| Cisco | 474 | 0.8% |
| Unknown | 452 | 0.7% |
| Freescale | 447 | 0.7% |
| Astaro | 395 | 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 three companies.

## WHY COMPANIES SUPPORT LINUX 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.
- Companies which are not in the information technology business can still find working with Linux beneficial. The 2.6.25 kernel included an implementation of the PF_CAN network protocol which was contributed by Volkswagen. 2.6.30 had a patch from Quantum Controls BV, which makes navigational devices for yachts. These companies find Linux to be a solid platform upon which to build their products; they contribute to the kernel to help ensure that Linux continues to meet their needs into the future. No other operating system gives this power to influence future development to its users.

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.

## 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. 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 Jonthan 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 **http://www.kernel.org/pub/linux/kernel/people/gregkh/kernel_history/**