

Linux Kernel Development

Greg Kroah-Hartman
gregkh@linuxfoundation.org

github.com/gregkh/kernel-development

58,000 files
23,100,000 lines

3,781 developers
≈400 companies

7,300 lines added
2,400 lines removed
2,000 lines modified

7,300 lines added
2,400 lines removed
2,000 lines modified

Every day

8 changes per hour

Kernel releases 4.7.0 – 4.11.0
May 2016 – April 2017

9.7 changes per hour

4.9 release

4.12 release July 7th?

2nd largest release

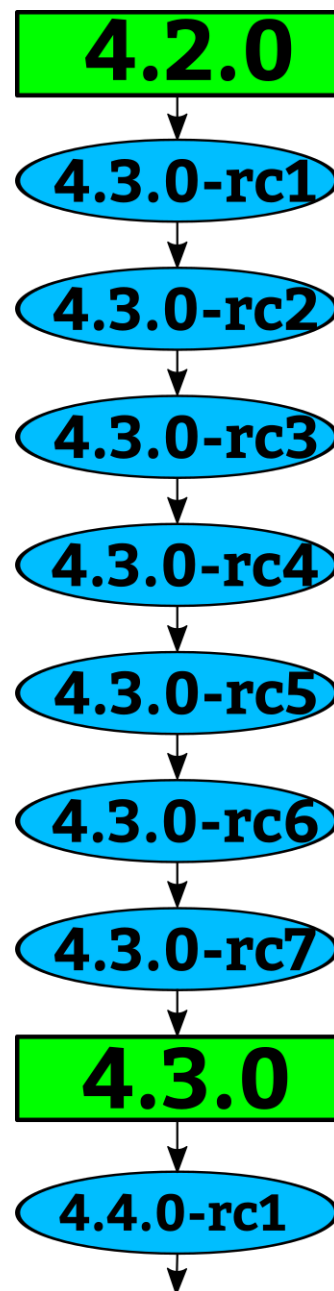
How we stay sane

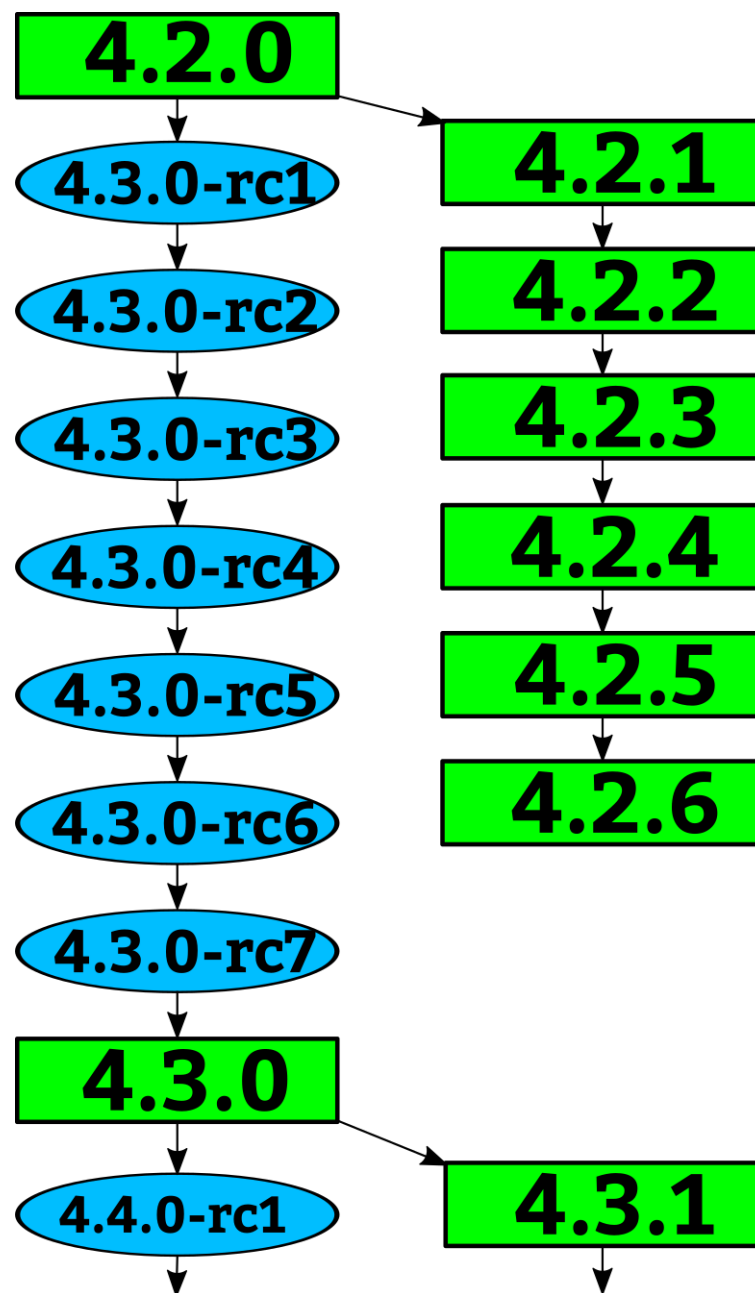
Time based releases

Incremental changes



**New release every
2½ months**





“Longterm kernels”

One picked per year

Maintained for two years

4.4

4.9

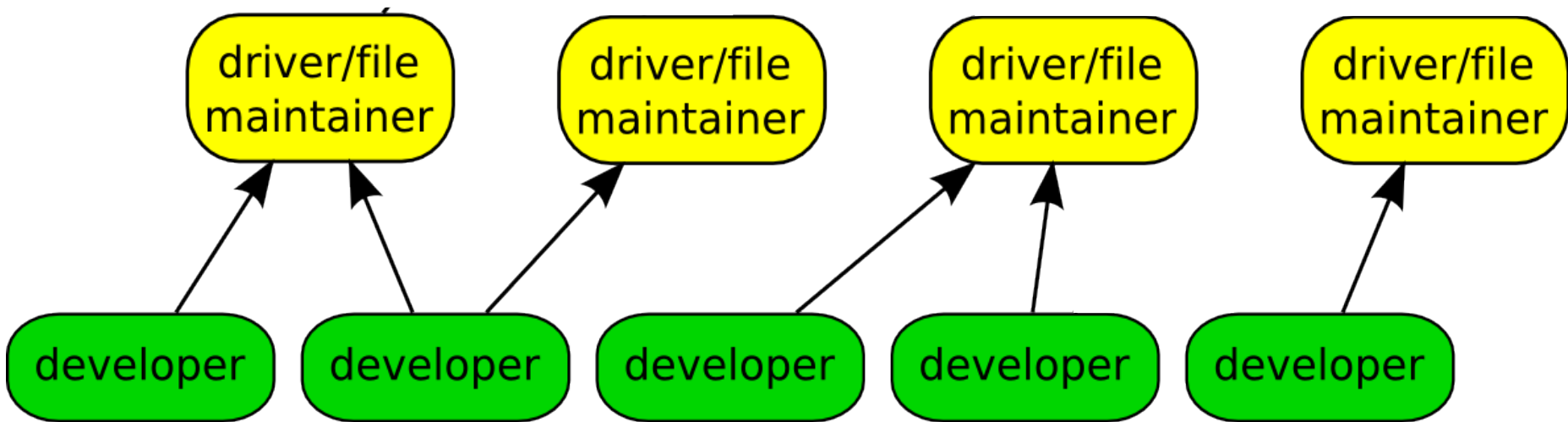
developer

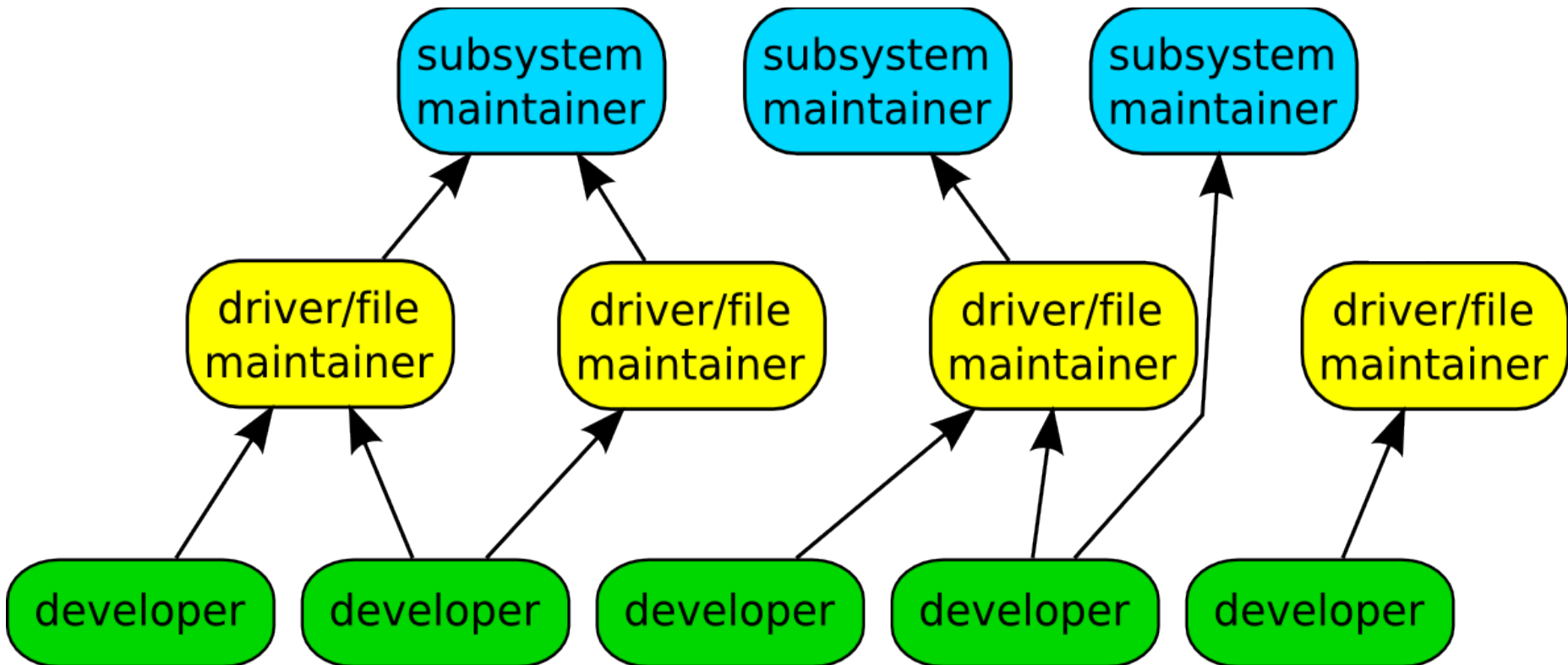
developer

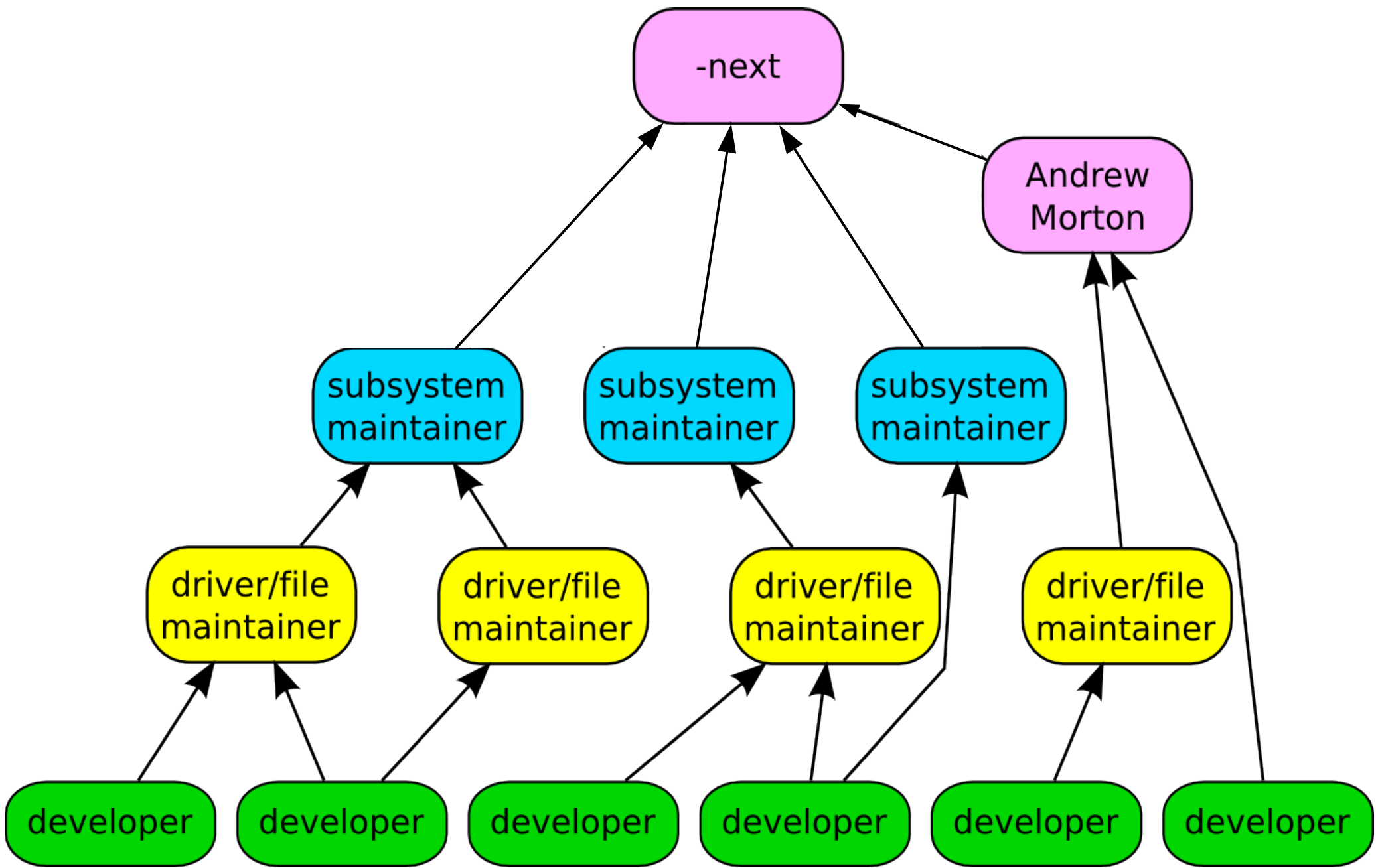
developer

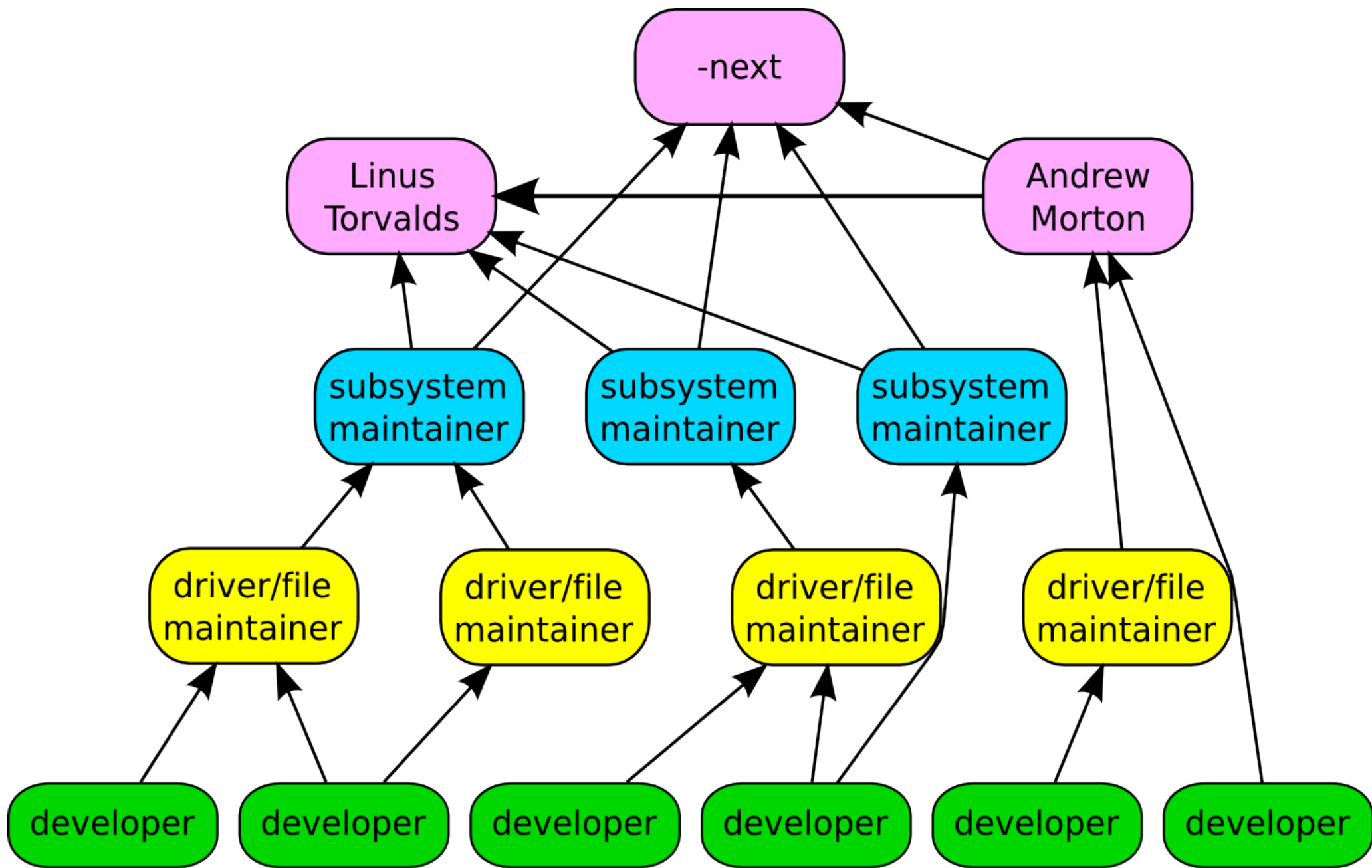
developer

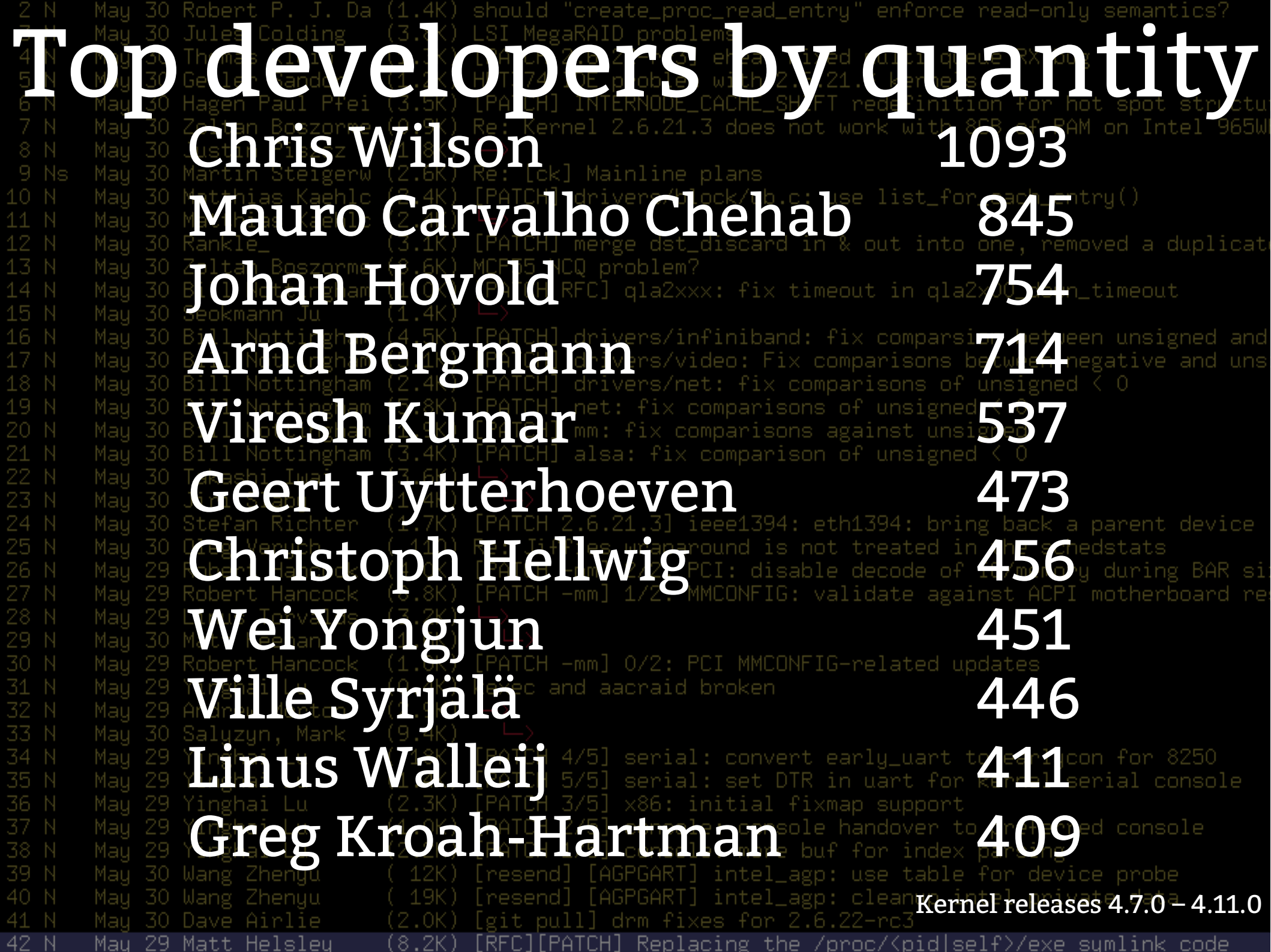
developer











Top Signed-off-by:

Greg Kroah-Hartman 7734

David S. Miller 7107

Mauro Carvalho Chehab 2317

Linus Torvalds 2144

Mark Brown 1966

Andrew Morton 1930

Ingo Molnar 1809

Alex Deucher 1529

Linus Walleij 1202

Chris Wilson 1199

Kalle Valo 1196

Kernel releases 4.7.0 – 4.11.0

Who is funding this work?

1. “Amateurs”	14.4%
2. Intel	13.4%
3. Red Hat	7.3%
4. Linaro	6.4%
5. IBM	3.4%
6. Samsung	3.4%
7. Consultants	3.0%
8. SuSE	2.9%
9. Google	2.7%
10. AMD	2.3%

Who is funding this work?

11. Mellanox	1.9%
12. Renesas Electronics	1.8%
13. Huawei Technologies	1.6%
14. Oracle	1.6%
15. Broadcom	1.5%
16. Texas Instruments	1.5%
17. ARM	1.4%
18. Free Electrons	1.1%
19. Imagination Technologies	1.0%
20. NXP Semiconductors	0.9%

“Working upstream saves time and money”

Dan Frye – VP Open Systems, IBM

Dirk Hohndel – Chief Technologist, Intel

How to change

How to change

Submit code early and often

How to change

Send small pieces

How to change

Ask community for feedback

How to change

Act on it

How to change

Remove legal hurdles

How to change

Force them to work in public

How to change

Allow them to be the community

Kernel Security

Kernel Security

Almost all bugs can be a “security” issue.

Kernel Security

Almost all bugs can be a “security” issue.

Fix them as soon as possible.

Kernel Security

Averaging 13 fixes per day.

“If you are not using a stable /
longterm kernel, your machine
is insecure”

– me

“The kernel needs airbags” – Konstantin Ryabitsev

slides.com/mricon/giant-bags-of-mostly-water#/

**“We will always have bugs,
we must stop their exploitation”
– Kees Cook**

outflux.net/slides/2015/ks/security.pdf

Kernel Hardening

kernsec.org/wiki/index.php/Kernel_Self_Protection_Project

Core Infrastructure Initiative

**“Ceaseless change is the only
constant thing in Nature.”**

– John Candee Dean



github.com/gregkh/kernel-development

Linux Kernel Development

Greg Kroah-Hartman
gregkh@linuxfoundation.org

github.com/gregkh/kernel-development



I'm going to discuss the how fast the kernel is moving, how we do it all, and how you can get involved.

58,000 files
23,100,000 lines

Kernel release 4.11.0

This was for the 4.11 kernel release, which happened April 30, 2017.

3,781 developers ≈400 companies

Kernel releases 4.7.0 – 4.11.0
May 2016 – April 2017

This makes the Linux kernel the largest contributed body of software out there that we know of.

This is just the number of companies that we know about, there are more that we do not, and as the responses to our inquiries come in, this number will go up.

Have surpassed 400 companies for 4 years now.

7,300 lines added
2,400 lines removed
2,000 lines modified

Kernel releases 4.7.0 – 4.11.0
May 2016 – April 2017

7,300 lines added
2,400 lines removed
2,000 lines modified

Every day

Kernel releases 4.7.0 – 4.11.0
May 2016 – April 2017

8 changes per hour

Kernel releases 4.7.0 – 4.11.0
May 2016 – April 2017

This is 24 hours a day, 7 days a week, for a full year.

We went this fast the year before this as well, this is an amazing rate of change.

Interesting note, all of these changes are all through the whole kernel.

For example, the core kernel is only 5% of the code, and 5% of the change was to the core kernel. Drivers are 55%, and 55% was done to them, it's completely proportional all across the whole kernel.

9.7 changes per hour

4.9 release

4.9 was the “largest” in number of changes that we have ever accepted. After 4.9, things went down a bit for 4.10 and 4.11, but 4.12 is getting very big.

Now this is just the patches we accepted, not all of the patches that have been submitted, lots of patches are rejected, as anyone who has ever tried to submit a patch can attest to.

4.12 release July 7th?

2nd largest release

4.12 should be released on July 7th and is on track to be the 2nd largest release by number of changes we have ever done.

And the first largest on number of lines of code we have added, due to some very large drivers being added to the tree.

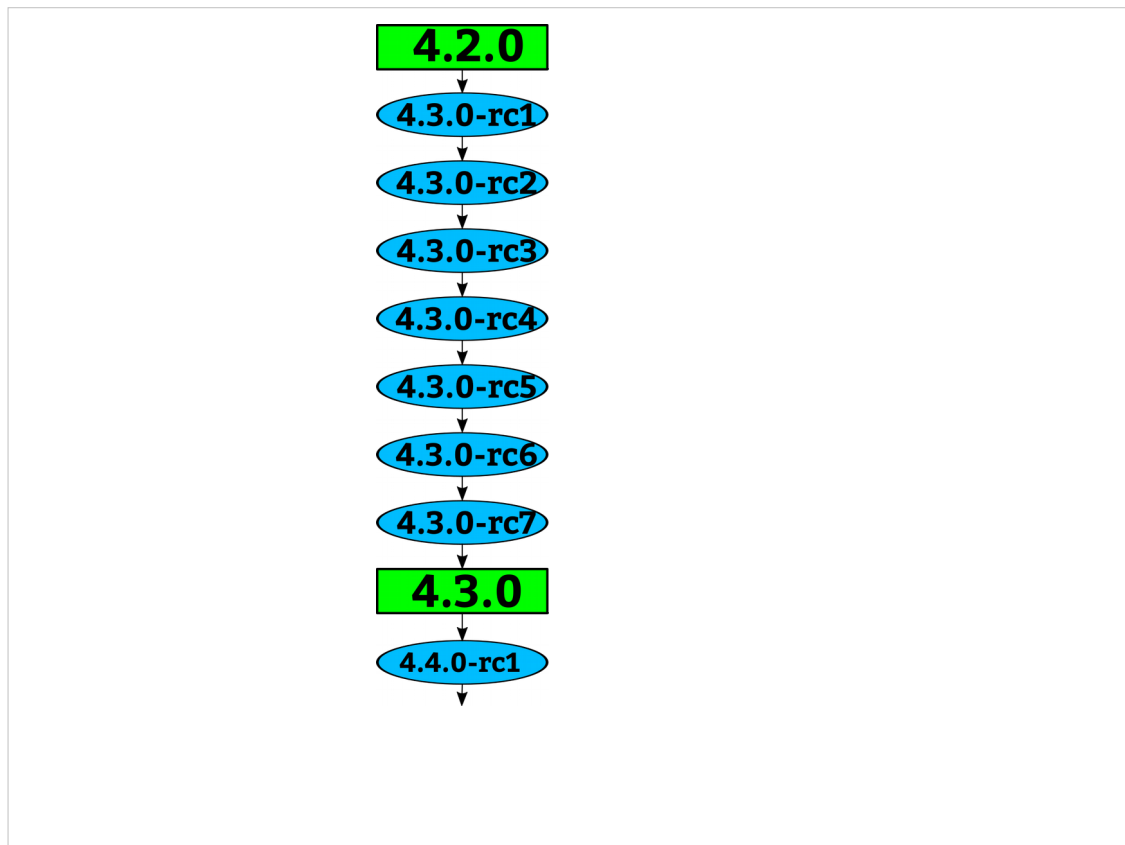
How we stay sane

Time based releases

Incremental changes



67 days to be exact, very regular experience.

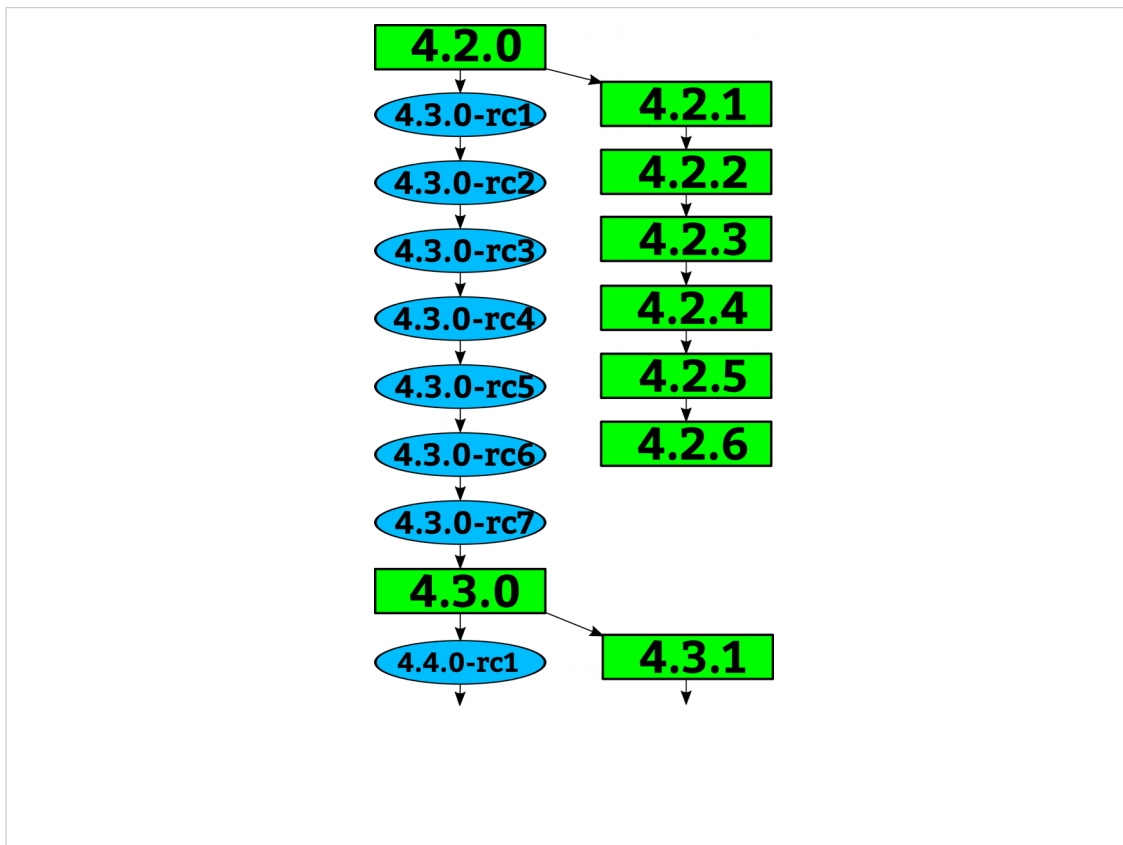


How a kernel is developed.

Linus releases a stable kernel

- 2 week merge window from subsystem maintainers
- rc1 is released
- bugfixes only now
- 2 weeks later, rc2
- bugfixes and regressions
- 2 weeks later, rc3

And so on until all major bugfixes and regressions are resolved and then the cycle starts over again.



Greg takes the stable releases from Linus, and does stable releases with them, applying only fixes that are already in Linus's tree.

Requiring fixes to be in Linus's tree first ensures that there is no divergence in the development model.

After Linus releases a new stable release, the old stable series is dropped.

With the exception of “longterm” stable releases, those are special, the stick around for much longer...

“Longterm kernels”

One picked per year
Maintained for two years

4.4 4.9

I pick one kernel release per year to maintain for longer than one release cycle. This kernel I will maintain for at least 2 years.

This means there are 2 longterm kernels being maintained at the same time.

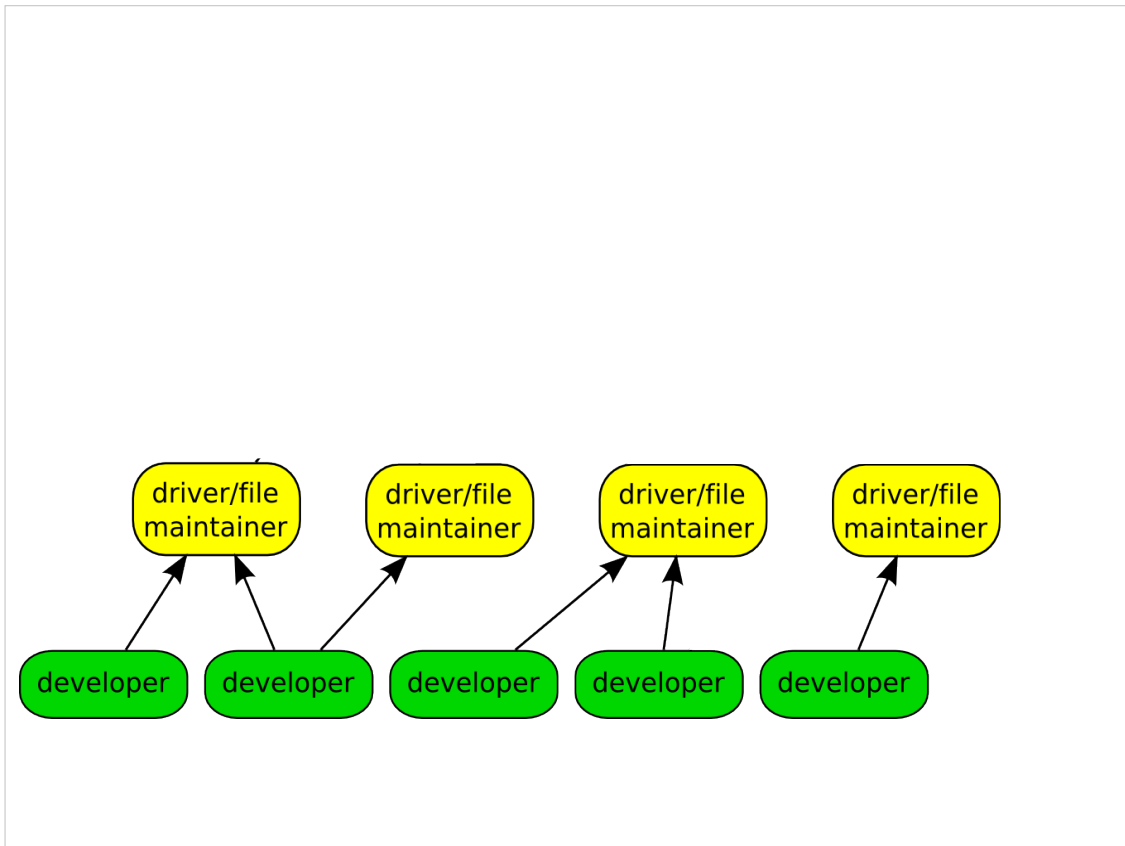
4.4 and 4.9 are the longterm kernel releases I am currently maintaining

The LTSI project is based on the longterm kernels.



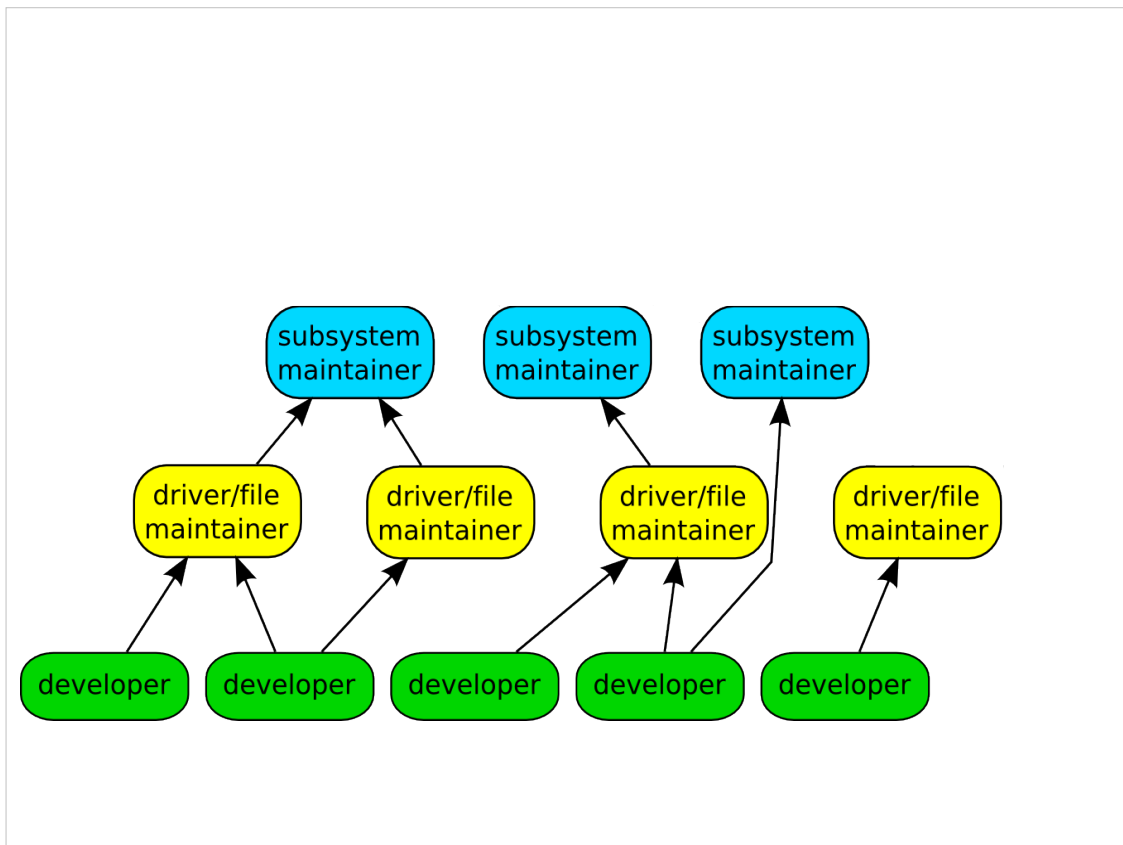
Like mentioned before, we have almost 3000 individual contributors. They all create a patch, a single change to the Linux kernel. This change could be something small, like a spelling correction, or something larger, like a whole new driver.

Every patch that is created only does one thing, and it can not break the build, complex changes to the kernel get broken up into smaller pieces.



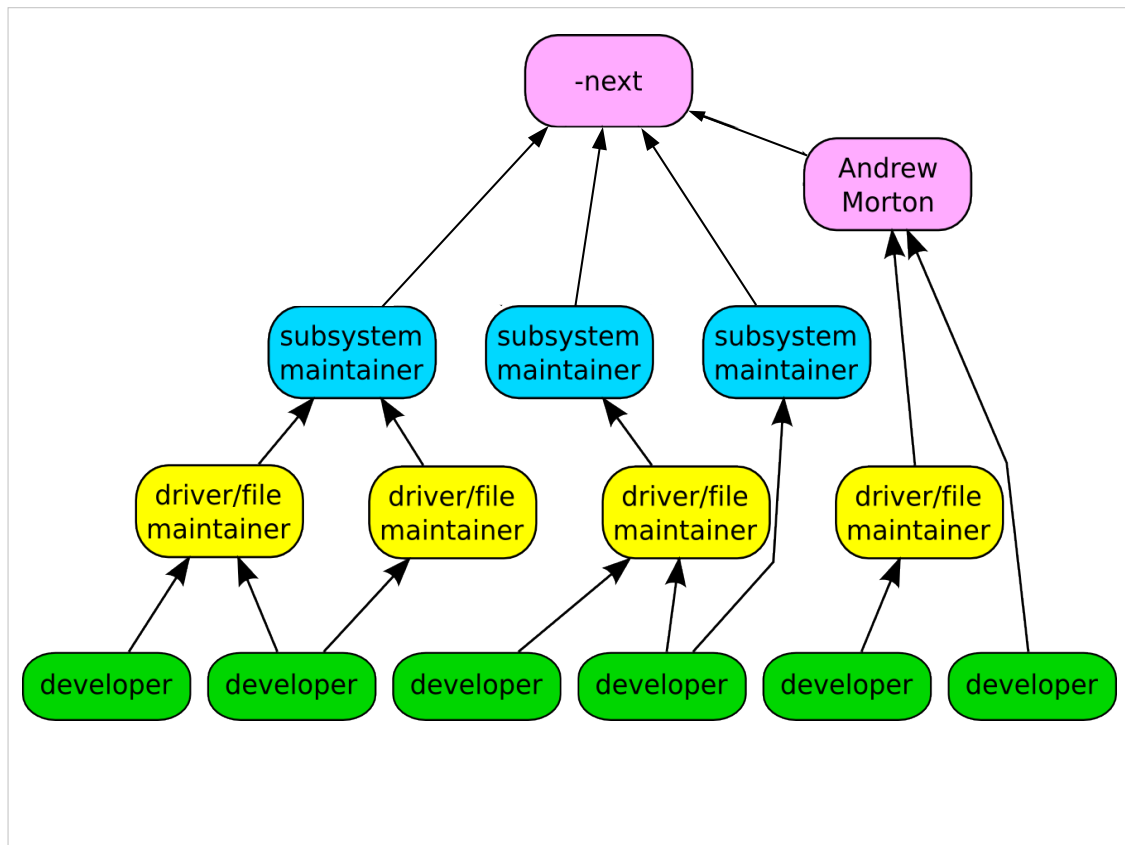
The developers send their patch to the maintainer of the file(s) that they have modified.

We have about 700 different driver/file/subsystem maintainers



After reviewing the code, and adding their own signed-off-by to the patch, the file/driver maintainer sends the patch to the subsystem maintainer responsible for that portion of the kernel.

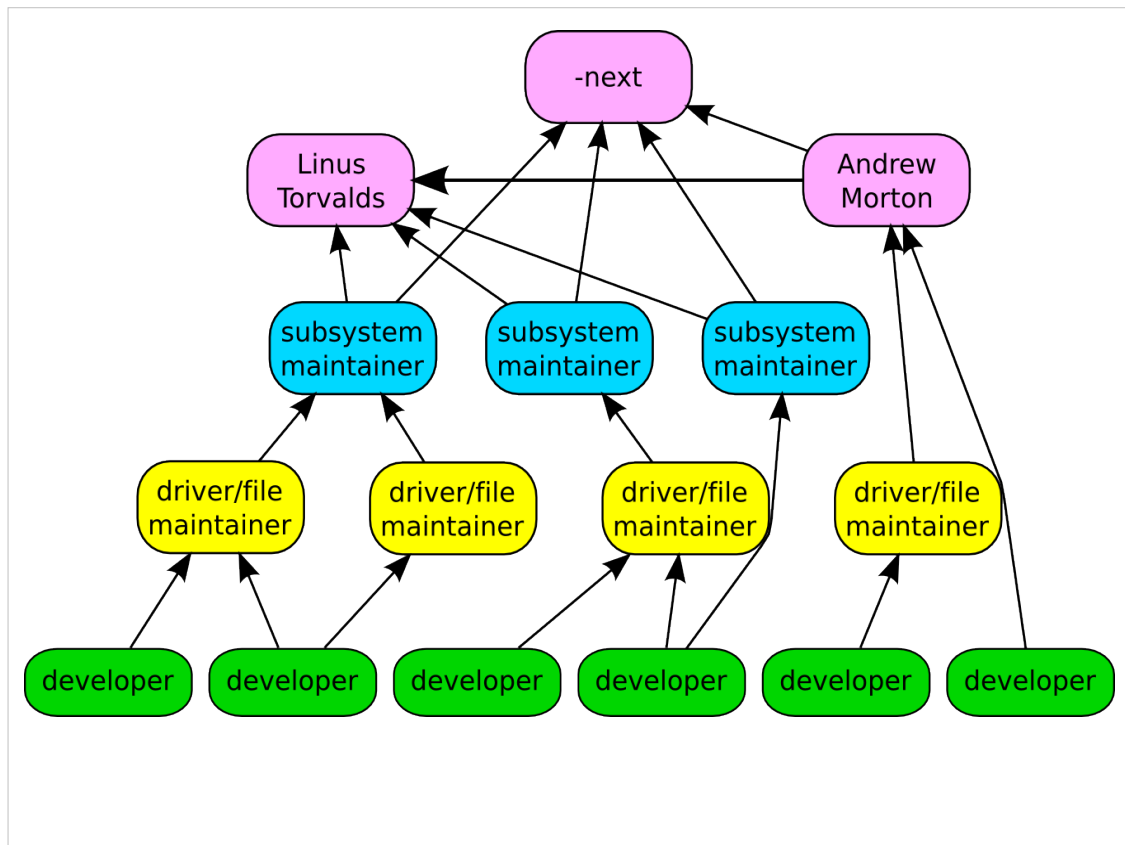
We have around 150 subsystem maintainers



Linux-next gets created every night from all of the different subsystem trees and build tested on a wide range of different platforms.

We have about 150 different trees in the linux-next release.

Andrew Morton picks up patches that cross subsystems, or are missed by others, and releases his -mm kernels every few weeks. This includes the linux-next release at that time.



Every 3 months, when the merge window opens up, everything gets sent to Linus from the subsystem maintainers and Andrew Morton.

The merge window is 2 weeks long, and thousands of patches get merged in that short time.

All of the patches merged to Linus should have been in the linux-next release, but that isn't always the case for various reasons.

Linux-next can not just be sent to Linus as there are things in there that sometimes are not good enough to be merged just yet, it is up to the individual subsystem maintainer to decide what to merge.

Top developers by quantity		
2 N	May 30 Robert P. J. Da (1.4K) should "create_proc_read_entry" enforce read-only semantics?	
3 N	May 30 Jules Colding (3.5K) LSI MegaRAID problem	
4 N	May 30 Thomas Gleixner (7.7K) [PATCH] drivers/usb/lcd: use list_for_each_entry()	
5 N	May 30 Geert Uytterhoeven (2.2K) [PATCH] drivers/usb/lcd: use list_for_each_entry()	
6 N	May 30 Hagen Paul Pfei (3.5K) [PATCH] INTERNUDE_CACHE_SHIFT read definition for hot spot statu	
7 N	May 30 2013 Jan Brzozowski (1.0) Re: Kernel 2.6.21.3 does not work with 8086 SAM on Intel 965W	
8 N	May 30 Chris Wilson (1.4K) Re: [ck] Mainline plans	1093
9 Ns	May 30 Martin Steigerw (2.6K) Re: [ck] Mainline plans	
10 N	May 30 Matthias Kaehle (1.4K) [PATCH] drivers/usb/lcd: use list_for_each_entry()	845
11 N	May 30 Mauro Carvalho Chehab (3.1K) [PATCH] merge dst_discard in & out into one, removed a duplicat	
12 N	May 30 Hankle (3.1K) [PATCH] merge dst_discard in & out into one, removed a duplicat	754
13 N	May 30 2013 Jan Brzozowski (1.0) Re: Kernel 2.6.21.3 does not work with 8086 SAM on Intel 965W	
14 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	714
15 N	May 30 Beckmann J (1.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
16 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	537
17 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
18 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	473
19 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
20 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	456
21 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
22 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	451
23 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
24 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	446
25 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
26 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	411
27 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
28 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	409
29 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
30 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
31 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
32 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
33 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
34 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
35 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
36 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
37 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
38 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
39 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
40 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
41 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	
42 N	May 30 Bill Nottingham (3.4K) [PATCH] qla2xxx: fix timeout in qla2xxx_timeout	

- Chris - intel graphics drivers
- Mauro - Video 4 Linux (media drivers)
- Johan - greybus, usb-serial, drivers
- Arnd - janitorial cleanups and arch-generic
- Viresh - greybus
- Geert - janitorial
- Christoph - vfs, filesystems, xfs, everywhere
- Wei - Janitorial
- Ville - intel graphics
- Linus - gpio, pin, arm drivers
- Greg - greybus

Top Signed-off-by:		
Greg Kroah-Hartman	7734	
David S. Miller	7107	
Mauro Carvalho Chehab	2317	
Linus Torvalds	2144	
Mark Brown	1966	
Andrew Morton	1930	
Ingo Molnar	1809	
Alex Deucher	1529	
Linus Walleij	1202	
Chris Wilson	1199	
Kalle Valo	1196	
Kernel releases 4.7.0 - 4.11.0		

Greg - driver core, usb, staging, greybus

David - networking, isa

Mauro - video 4 linux (media)

Linus - everything

Mark - embedded sound

Andrew - everything

Ingo - x86

Alex - radeon graphics

Linus - gpio and pinctl

Chris - intel graphics

Kalle - wireless drivers

Who is funding this work?

1. "Amateurs"	14.4%
2. Intel	13.4%
3. Red Hat	7.3%
4. Linaro	6.4%
5. IBM	3.4%
6. Samsung	3.4%
7. Consultants	3.0%
8. SuSE	2.9%
9. Google	2.7%
10. AMD	2.3%

Kernel releases 4.7 – 4.11

So you can view this as either 14% is done by non-affiliated people, or 86% is done by companies.

Now to be fair, if you show any skill in kernel development you are instantly hired.

Why this all matters: If your company relies on Linux, and it depends on the future of Linux supporting your needs, then you either trust these other companies are developing Linux in ways that will benefit you, or you need to get involved to make sure Linux works properly for your workloads and needs.

Who is funding this work?

11. Mellanox	1.9%
12. Renesas Electronics	1.8%
13. Huawei Technologies	1.6%
14. Oracle	1.6%
15. Broadcom	1.5%
16. Texas Instruments	1.5%
17. ARM	1.4%
18. Free Electrons	1.1%
19. Imagination Technologies	1.0%
20. NXP Semiconductors	0.9%

Kernel releases 4.7 – 4.11

Intel – 9000 patches

Huawei - 1115 patches (almost half done by
one developer!!!)

NXP - 636

“Working upstream saves time and money”

Dan Frye – VP Open Systems, IBM

Dirk Hohndel – Chief Technologist, Intel

How to change

“Change or die”

How to change

Submit code early and often

No big code dumps.

They are hard to review, and even harder for you to modify and resend, slowing everything down and delaying any potential acceptance.

How to change

Send small pieces

How to change

Ask community for feedback

Some companies get Linux kernel community members together and discuss products and technologies directly with the senior engineers, no managers in the way.

Great feedback circle, the community gets to understand your products better, and no long explanation is needed when showing the code later, and the community gets to tell your engineers what they are doing wrong.

How to change

Act on it

Change products / roadmaps / features based on feedback. This makes Linux work better on your platforms which makes your platform better.

How to change

Remove legal hurdles

Let them contribute whatever they want and can.

Legal is to support the business, change the hurdles to be on the legal side, not the developers.

How to change

Force them to work in public

No internal mailing lists

All communication is done publicly

Let them argue in public

How to change

Allow them to be the community

Your developers will become the maintainers, driving the future of Linux forward, always keeping your products in mind as things evolve and change.

Kernel Security

Let's talk about kernel security.

Kernel Security

Almost all bugs can be a “security” issue.

Anything that goes wrong in the kernel can usually be turned into a “security” problem.

Be it a DoS, or a reboot, or local root exploit, or worst case, a remote root exploit (very rare, thankfully.)

Kernel Security

Almost all bugs can be a “security” issue.

Fix them as soon as possible.

Because it's really hard to determine if a bug is a “security” issue, our response is that we fix all bugs as soon as possible once we learn about them.

TTY bug in RH

Kernel Security

Averaging 13 fixes per day.

If you look at the number of patches flowing into the stable tree, we are averaging 13 patches a day, every single day.

Now not all of them are “security” fixes. But some small percentage is.

This is for the latest kernel release, the 4.4 kernel is averaging 9 fixes a day, and 4.9 is still running at 13 fixes a day!

**“If you are not using a stable /
longterm kernel, your machine
is insecure”**

– me

Your infrastructure HAS to support updating the kernel. If you can't do that, you are insecure.

Even the “enterprise” kernels aren't keeping up with this rate of change, the exception being Debian.

If you use these kernels, you HAVE to keep up to date.

Android example.

“The kernel needs airbags”

– Konstantin Ryabitsev

slides.com/mricon/giant-bags-of-mostly-water#/

kernel.org sysadmin, in charge of the LF sysadmin team, Fedora infrastructure developer.

Great presentation on how you, as a sysadmin, can implement secure practices for your network. Full checklist and guide has been published.

But, even with those practices, we need low-level changes in order to save ourselves from the accidents that will happen.

We need “airbags” in the kernel, and elsewhere.

Things like SELinux, grsec, openwall we need them.

**“We will always have bugs,
we must stop their exploitation”
– Kees Cook**

outflux.net/slides/2015/ks/security.pdf

Kees Cook, kernel security developer, presentation at kernel summit last year.

We need to start doing things to make the kernel more “robust” from a security standpoint.

Even if it makes things harder for the developers.

Everyone agreed.

Kernel Hardening

kernsec.org/wiki/index.php/Kernel_Self_Protection_Project

Core Infrastructure Initiative

Kernel hardening project.

New security features are being added in each release, but if you don't upgrade, you don't get those features, and protection.

CII is helping to fund this, if you want to work on it, we need developers, and we will pay for it.

**“Ceaseless change is the only
constant thing in Nature.”**

– John Candee Dean

1911 astronomer.

If your operating system isn't constantly changing,
then it is dead. The world doesn't stop changing,
learn to embrace the change in order to survive.

“static systems” die.



github.com/gregkh/kernel-development

Obligatory Penguin Picture

