Linux Kernel Development

Greg Kroah-Hartman
gregkh@linuxfoundation.org

github.com/gregkh/kernel-development

47,000 files
18,900,000 lines

Kernel release 3.16.0

3,483 developers
439 companies

Kernel releases 3.11.0 – 3.16.0
September 2013 – August 2014

8,700 lines added
3,880 lines removed
1,900 lines modified

Kernel releases 3.11.0 – 3.16.0
September 2013 – August 2014

8,700 lines added
3,880 lines removed
1,900 lines modified

Every day

Kernel releases 3.11.0 – 3.16.0
September 2013 – August 2014

7.6 changes per hour

Kernel releases 3.11.0 – 3.16.0
September 2013 – August 2014

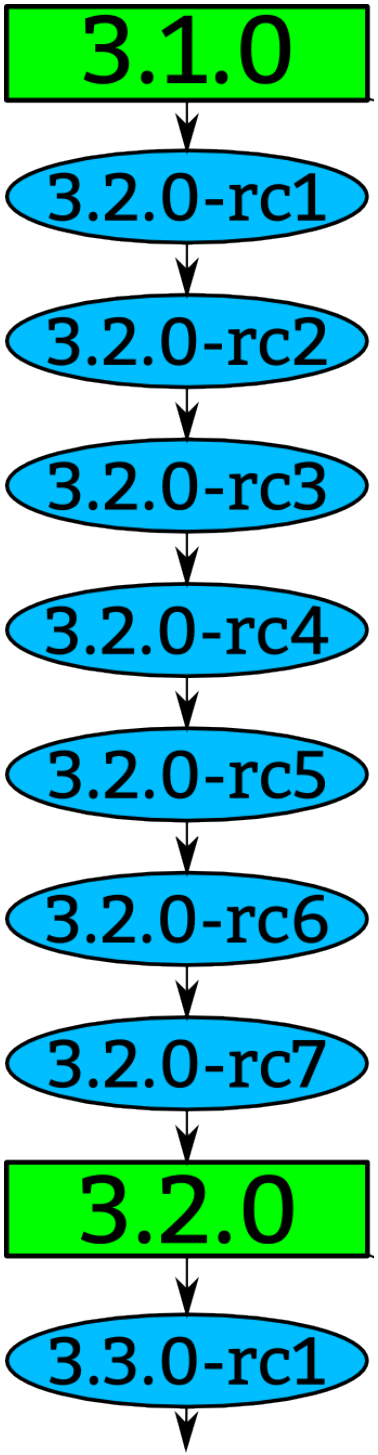
9.5 changes per hour

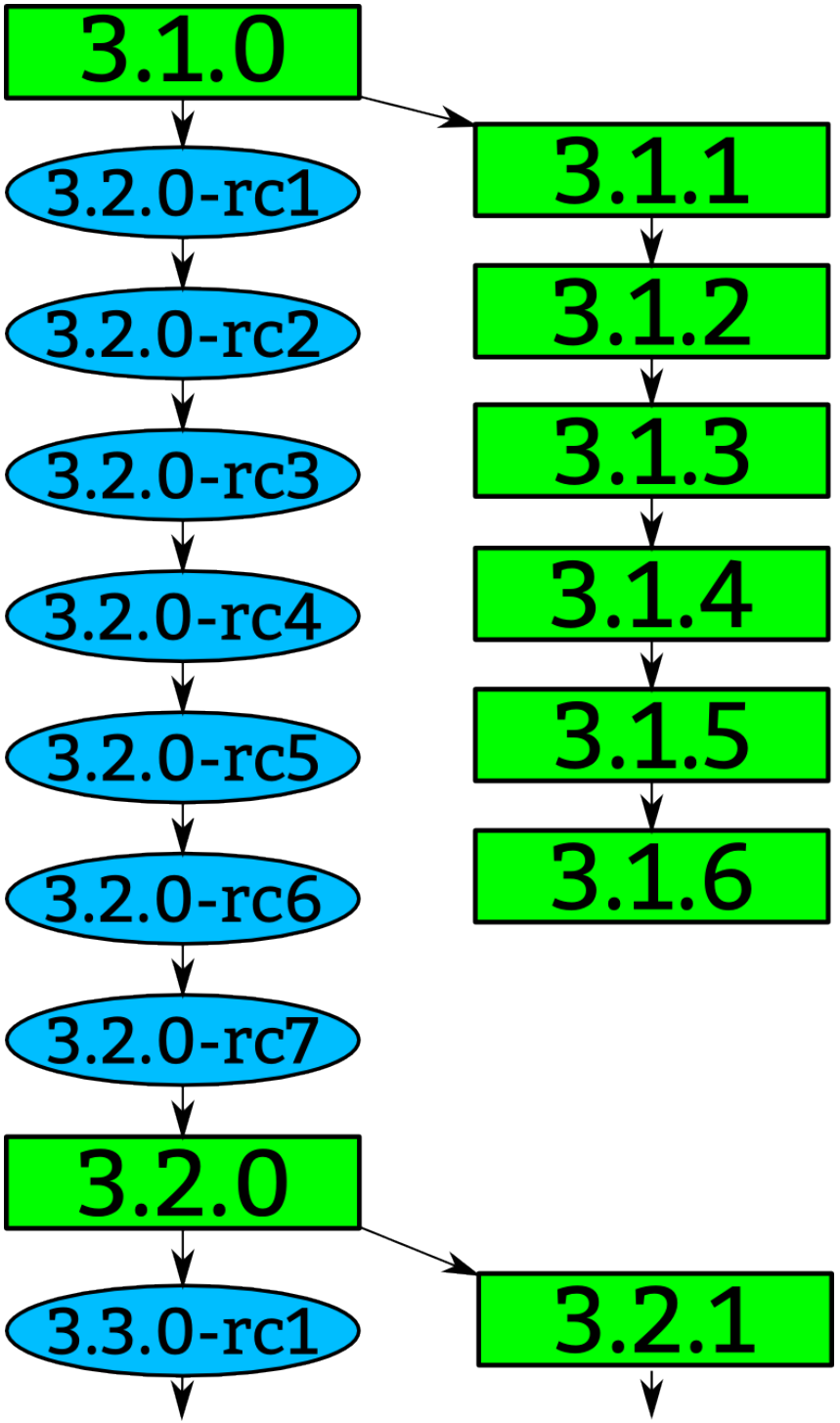
3.16 release

How we stay sane

Time based releases
Incremental changes


New release every
2½ months
Kernel releases 3.10.0 – 3.14.0


3.1.0
3.2.0-rc1
3.2.0-rc2
3.2.0-rc3
3.2.0-rc4
3.2.0-rc5
3.2.0-rc6
3.2.0-rc7
3.2.0
3.3.0-rc1

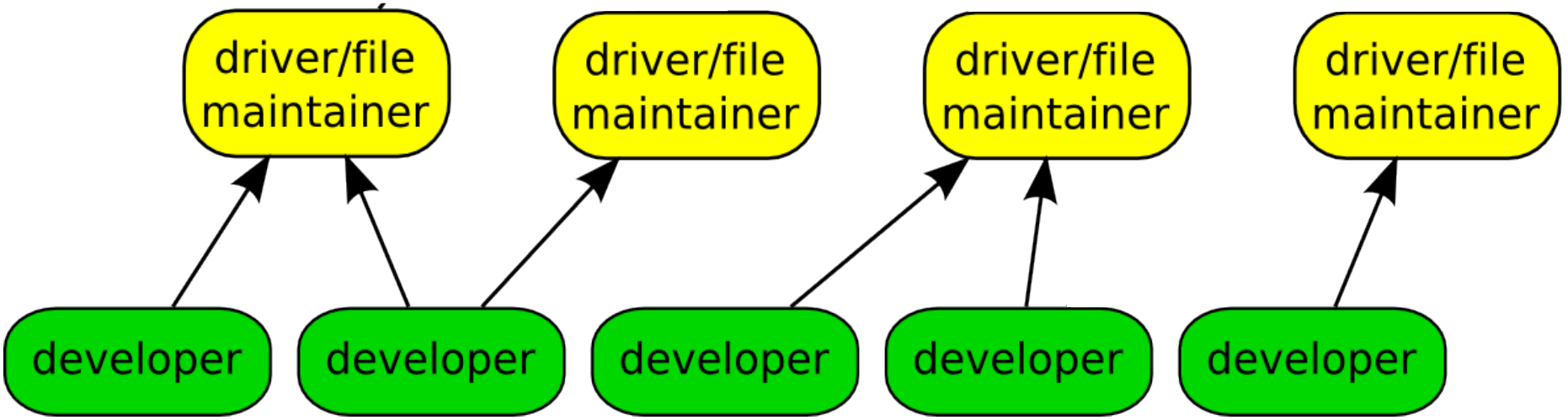

3.1.0
3.2.0-rc1
3.2.0-rc2
3.2.0-rc3
3.2.0-rc4
3.2.0-rc5
3.2.0-rc6
3.2.0-rc7
3.2.0
3.3.0-rc1
3.1.1
3.1.2
3.1.3
3.1.4
3.1.5
3.1.6
3.2.1

“Longterm kernels”

One picked per year
Maintained for two years

3.10 3.14


developer
developer
developer
developer
developer


driver/file maintainer
driver/file maintainer
driver/file maintainer
driver/file maintainer
developer
developer
developer
developer
developer

```
commit ecf85e481a716cfe07406439fdc7ba9526bbfaeb
Author:     Robert Jarzmik <robert.jarzmik@free.fr>
AuthorDate: Tue Apr 21 20:33:10 2009 -0700
Commit:     Greg Kroah-Hartman <gregkh@suse.de>
CommitDate: Thu Apr 23 14:15:31 2009 -0700

    USB: otg: Fix bug on remove path without transceiver

    In the case where a gadget driver is removed while no
    transceiver was found at probe time, a bug in
    otg_put_transceiver() will trigger.

    Signed-off-by: Robert Jarzmik <robert.jarzmik@free.fr>
    Acked-by: David Brownell <dbrownell@users.sourceforge.net>
    Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>

--- a/drivers/usb/otg/otg.c
+++ b/drivers/usb/otg/otg.c
@@ -43,7 +43,8 @@ EXPORT_SYMBOL(otg_get_transceiver);
 void otg_put_transceiver(struct otg_transceiver *x)
 {
-       put_device(x->dev);
+       if (x)
+               put_device(x->dev);
 }
```

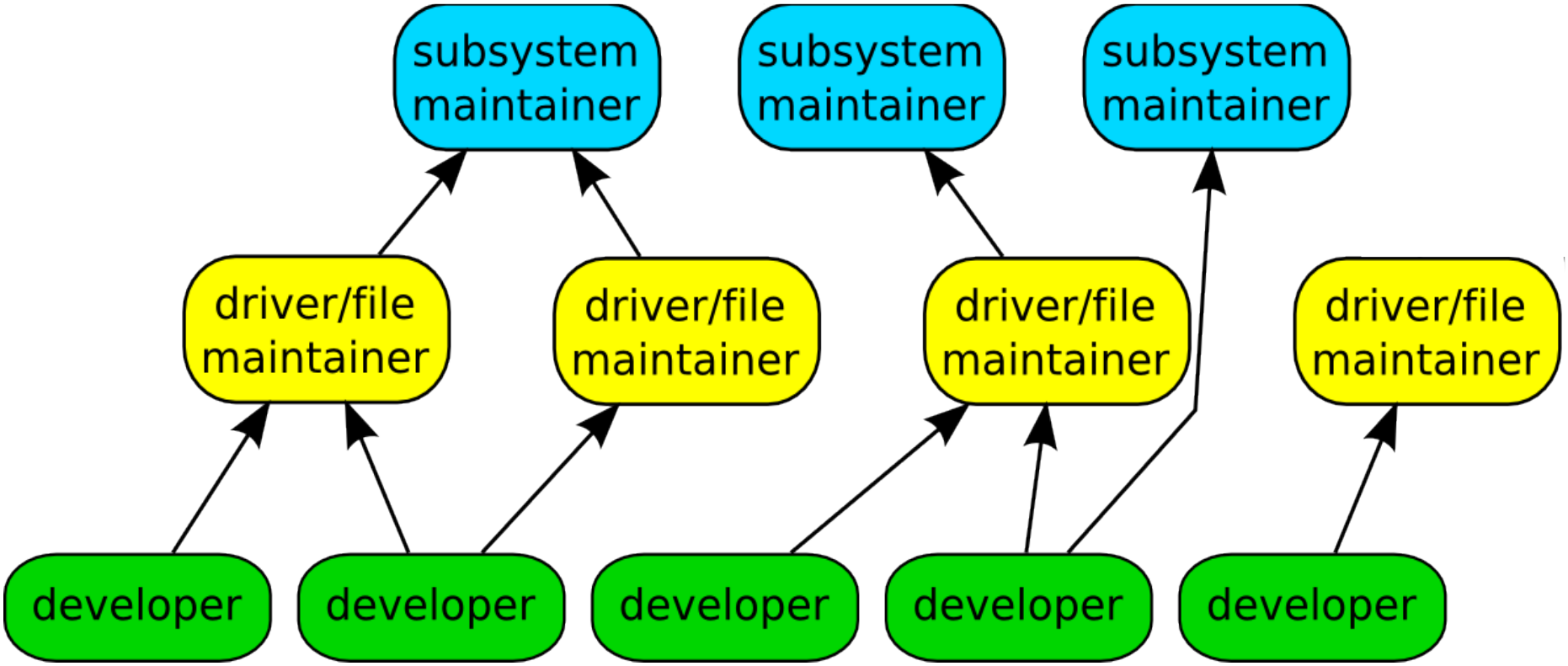
Developer's Certificate of Origin

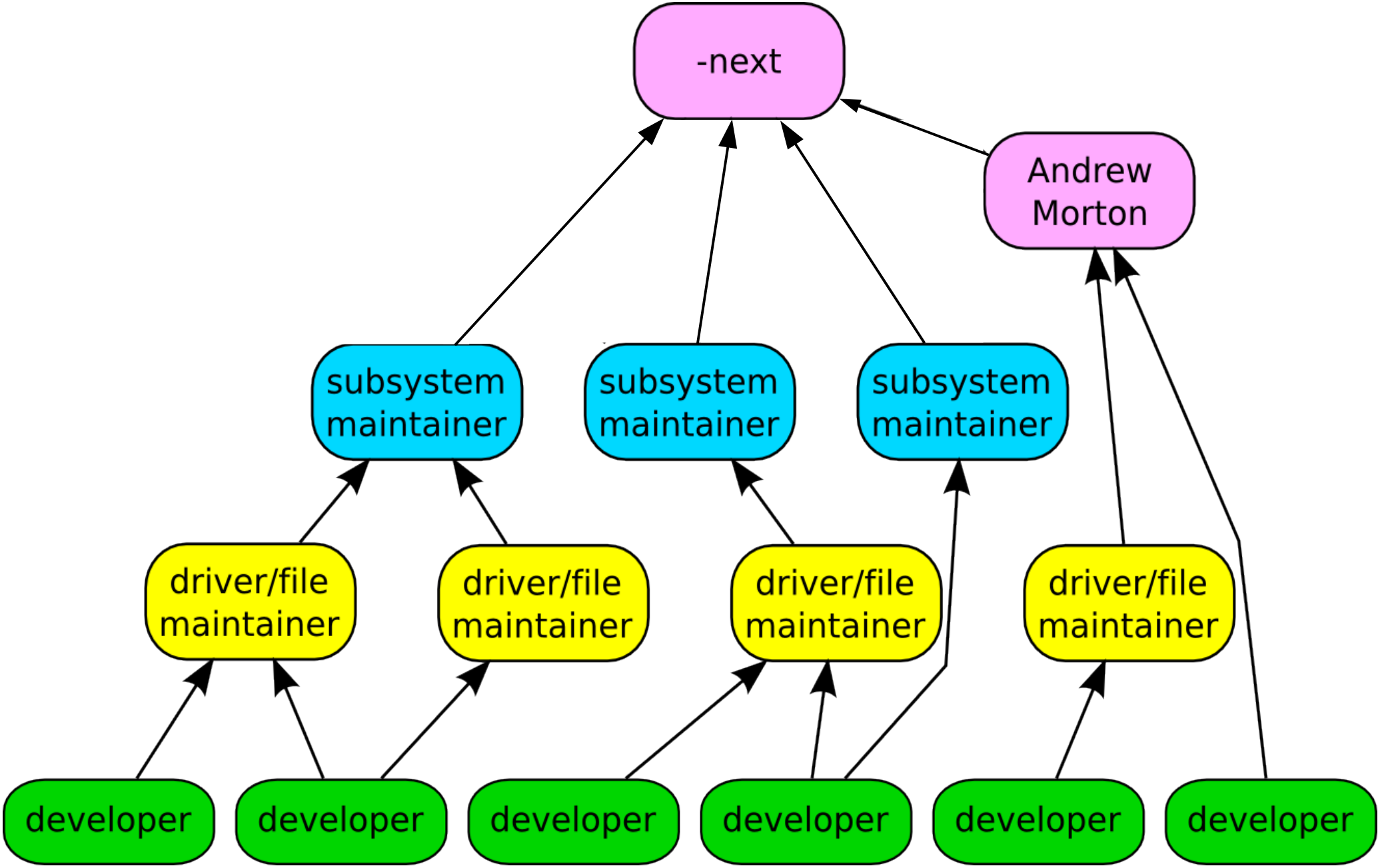
(a) I created this change; or

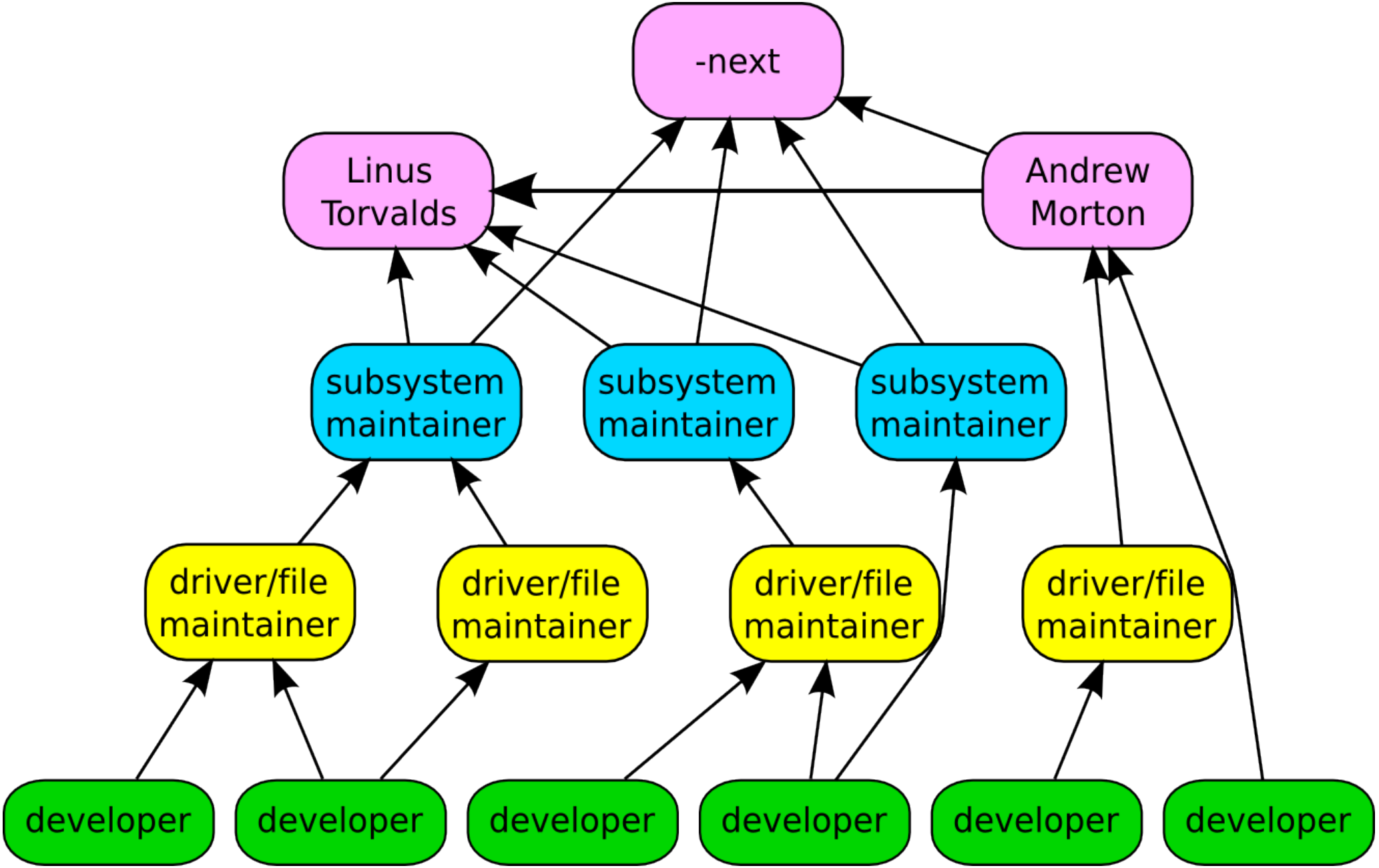
(b) Based this on a previous work with a compatible license; or

(c) Provided to me by (a), (b), or (c) and not modified

(d) This contribution is public.


subsystem maintainer
subsystem maintainer
subsystem maintainer
driver/file maintainer
driver/file maintainer
driver/file maintainer
driver/file maintainer
developer
developer
developer
developer
developer


-next
Andrew Morton
subsystem maintainer
subsystem maintainer
subsystem maintainer
driver/file maintainer
driver/file maintainer
driver/file maintainer
driver/file maintainer
developer
developer
developer
developer
developer
developer


-next
Linus Torvalds
Andrew Morton
subsystem maintainer
subsystem maintainer
subsystem maintainer
driver/file maintainer
driver/file maintainer
driver/file maintainer
driver/file maintainer
developer
developer
developer
developer
developer
developer

Top developers by quantity

H. Hartley Sweeten	1459
Sachin Kamat	1288
Jingoo Han	1167
Laurent Pinchart	820
Alex Deucher	642
Daniel Vetter	633
Lars-Peter Clausen	566
Mark Brown	537
Jes Sorensen	530
Ville Syrjälä	527

Kernel releases 3.11.0 – 3.16.0

Top Signed-off-by:

Greg Kroah-Hartman	9250
David S. Miller	5882
Mark Brown	3449
Linus Torvalds	3124
Andrew Morton	2963
Daniel Vetter	2649
Mauro Carvalho Chehab	2176
John Linville	1707
Rafael Wysocki	1606
H Hartley Sweeten	1459

Kernel releases 3.11.0 – 3.16.0

Who is funding this work?

1. Intel	10.6%
2. “Amateurs”	10.1%
3. Red Hat	8.4%
4. Linaro	6.3%
5. Unknown Individuals	5.6%
6. Samsung	4.5%
7. IBM	3.3%
8. SuSE	3.0%
9. Texas Instruments	2.5%
10. Consultants	2.5%

Kernel releases 3.11.0 – 3.16.0

Who is funding this work?

11. Google	2.0%
12. Vision Engraving	2.0%
13. Renesas	2.0%
14. Freescale	1.7%
15. Free Electrons	1.6%
16. FOSS OPFW	1.4%
17. Oracle	1.2%
18. AMD	1.2%
19. Nvidia	1.2%
20. Huawei	1.2%

Kernel releases 3.10.0 – 3.14.0

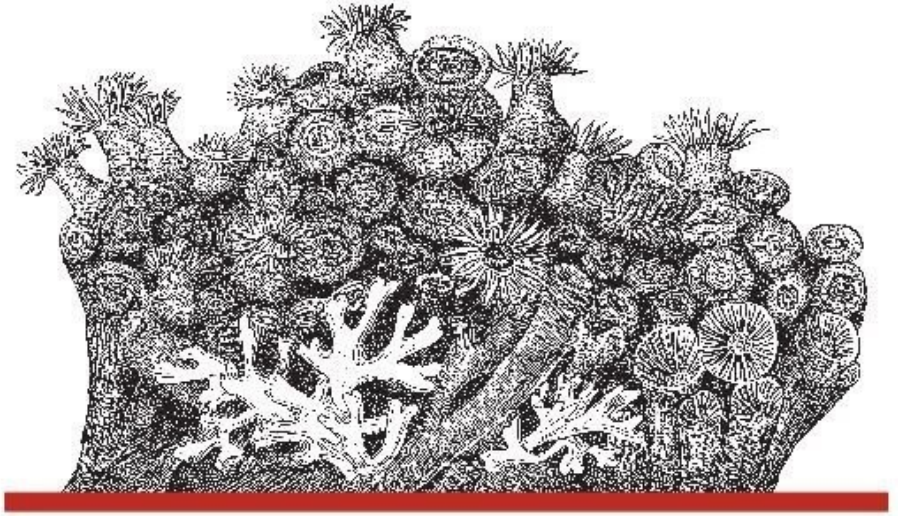
“Working upstream saves time and money”

Dan Frye – VP Open Systems, IBM
Dirk Hohndel – Chief Technologist, Intel

Getting involved

Run the kernel.org release on your machine

Getting involved



LINUX
KERNEL
IN A NUTSHELL

A Desktop Quick Reference

Getting involved

`Documentation/HOWTO`

`Documentation/development-process`

Getting involved

kernelnewbies.org

Getting involved

Google “write your first kernel patch”

Getting involved

kernelnewbies.org/KernelJanitors/Todo

Getting involved

Eudyptula Challenge
(little penguin)

http://eudyptula-challenge.org/

Getting involved

Linux Driver Project

`drivers/staging/*/TODO`

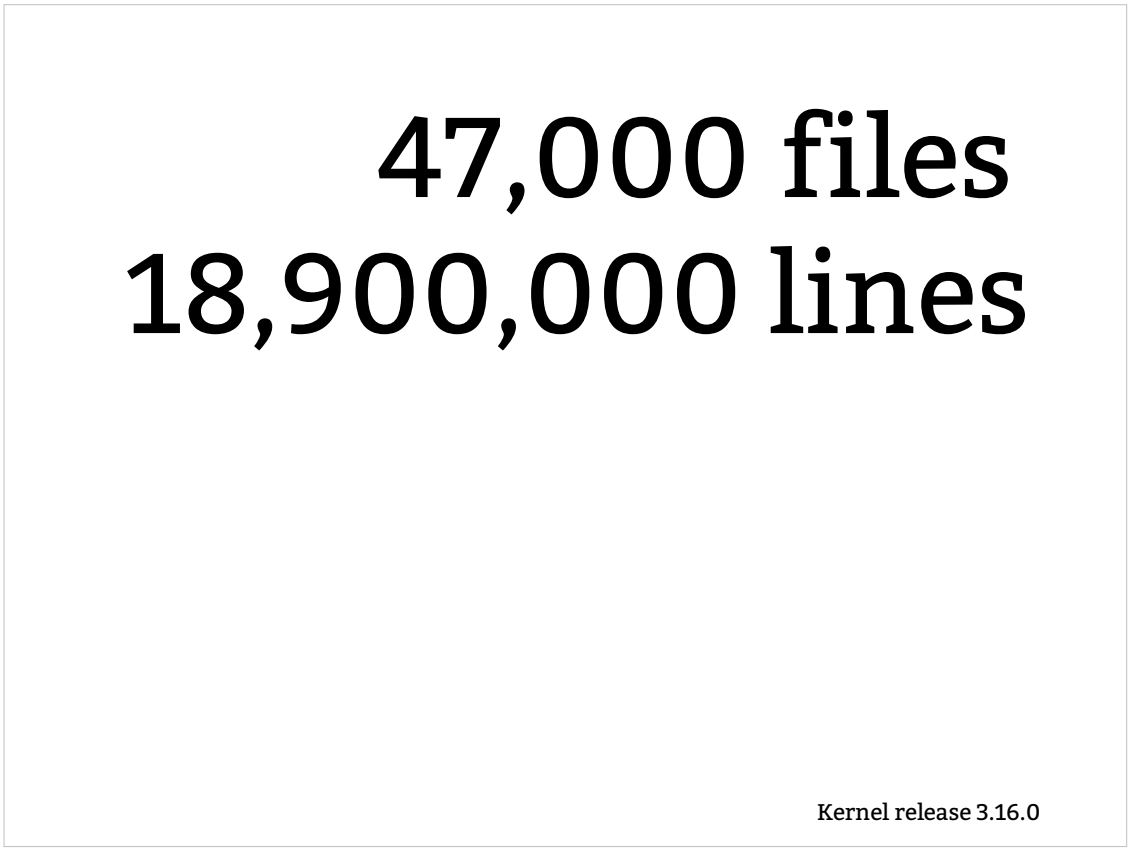
github.com/gregkh/kernel-development

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.



This was for the 3.16 kernel release, which happened August 3, 2014.

3,483 developers
439 companies

Kernel releases 3.11.0 – 3.16.0
September 2013 – August 2014

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 2 years now.

8,700 lines added
3,880 lines removed
1,900 lines modified

Kernel releases 3.11.0 – 3.16.0
September 2013 – August 2014

8,700 lines added
3,880 lines removed
1,900 lines modified

Every day

Kernel releases 3.11.0 – 3.16.0
September 2013 – August 2014

7.6 changes per hour

Kernel releases 3.11.0 – 3.16.0
September 2013 – August 2014

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.

9.5 changes per hour

3.16 release

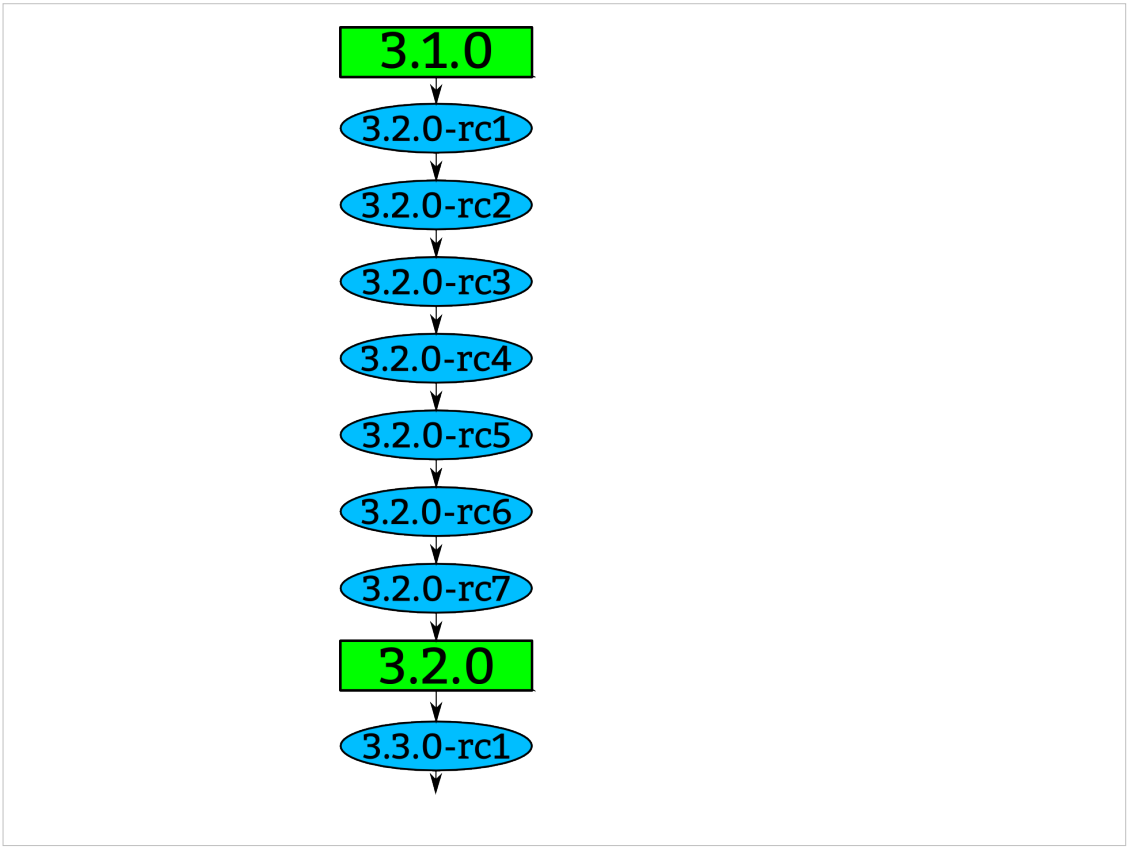
This past 3.16 release was the fastest we have ever created. That number shows just how well the Linux kernel development model is working. We are growing in developers and in how fast we are developing overall.

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.

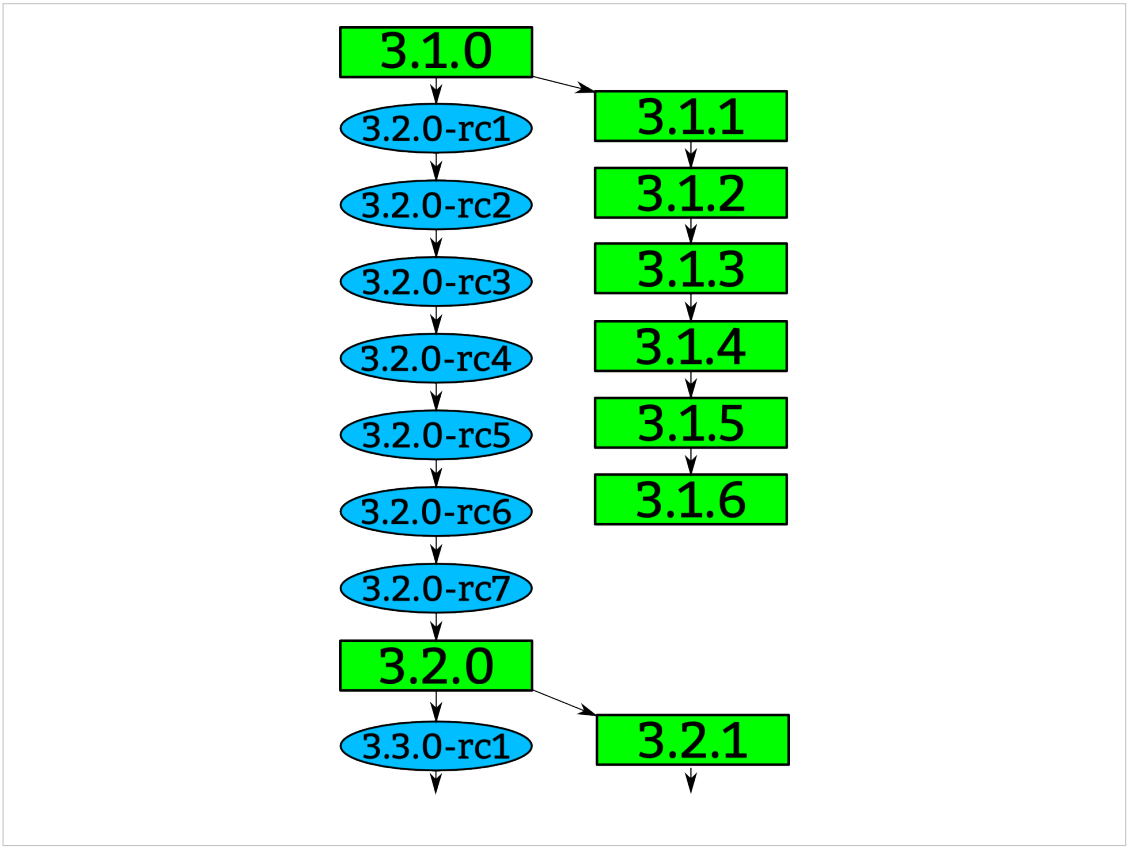
How we stay sane

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

3.10 3.14

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.

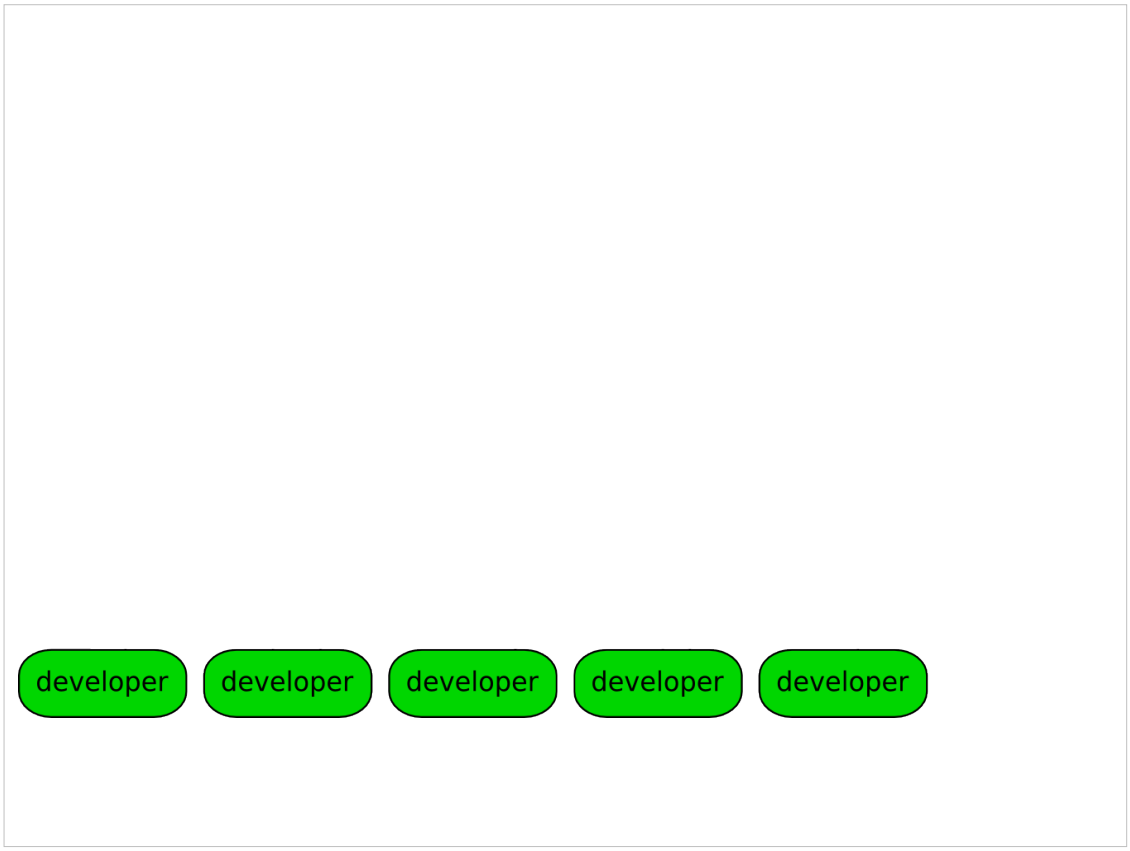
This means there are 2 longterm kernels being maintained at the same time.

3.4 and 3.10 are the longterm kernel releases I am maintaining.

3.4 will stop being maintained in October.

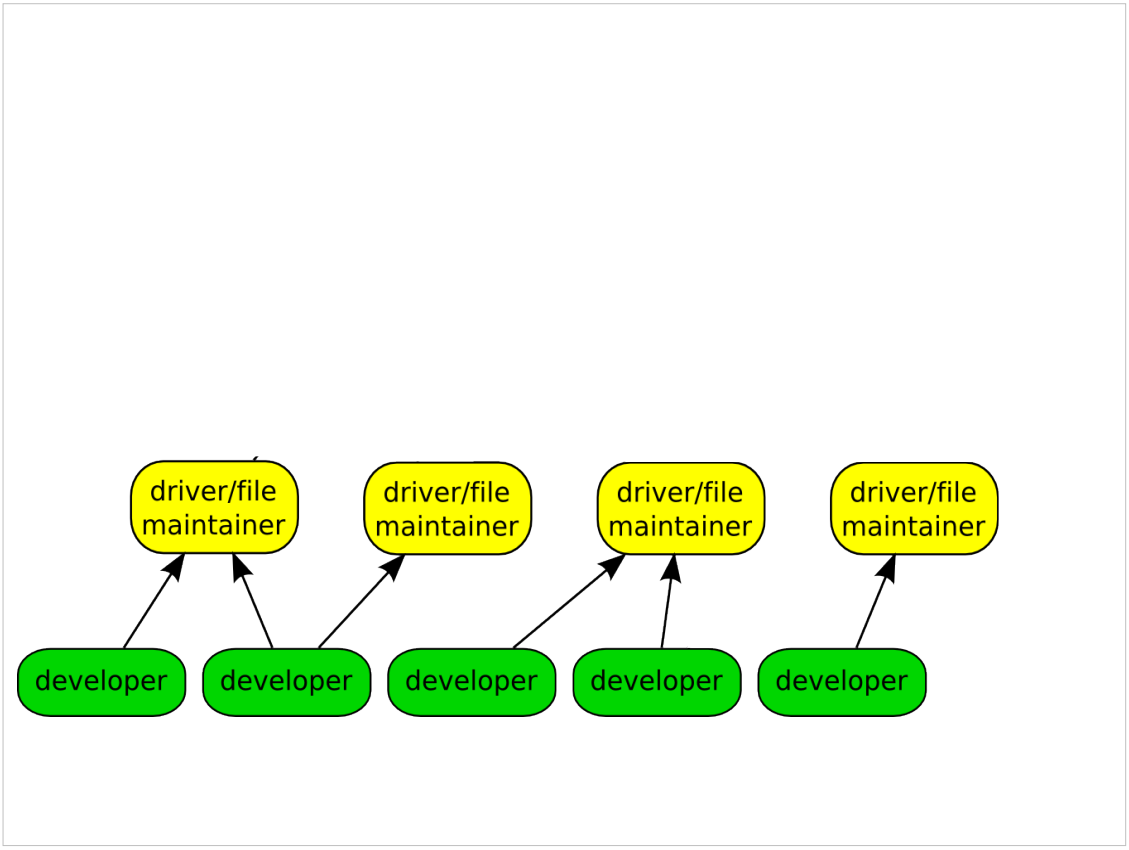
Ben Hutchings is maintaining the 3.2 kernel as a longterm kernel for the Debian project.

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

```
commit ecf85e481a716cfe07406439fdc7ba9526bbfaeb
Author:     Robert Jarzmik <robert.jarzmik@free.fr>
AuthorDate: Tue Apr 21 20:33:10 2009 -0700
Commit:     Greg Kroah-Hartman <gregkh@suse.de>
CommitDate: Thu Apr 23 14:15:31 2009 -0700

    USB: otg: Fix bug on remove path without transceiver

    In the case where a gadget driver is removed while no
    transceiver was found at probe time, a bug in
    otg_put_transceiver() will trigger.

    Signed-off-by: Robert Jarzmik <robert.jarzmik@free.fr>
    Acked-by: David Brownell <dbrownell@users.sourceforge.net>
    Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>

--- a/drivers/usb/otg/otg.c
+++ b/drivers/usb/otg/otg.c
@@ -43,7 +43,8 @@ EXPORT_SYMBOL(otg_get_transceiver);
 void otg_put_transceiver(struct otg_transceiver *x)
 {
-       put_device(x->dev);
+       if (x)
+               put_device(x->dev);
 }
```

This is an example of a patch.

It came from Robert, was acked by David, the maintainer at the time of the usb on-the-go subsystem, and then signed off by by me before it was commited to the kernel tree.

The change did one thing, it checked the value of the pointer before it was dereferenced, fixing a bug that would have crashed the kernel if it had been hit.

This is also a “blame” trail, showing who changed each line in the kernel, and who agreed with that change.

If a problem is found, these are the developers that you can ask about it.

Because of this, every line in the Linux kernel can be traced back to at least two developers who are responsible for it.

This is better than any other body of code.

Developer's Certificate of Origin

(a) I created this change; or

(b) Based this on a previous work with a compatible license; or

(c) Provided to me by (a), (b), or (c) and not modified

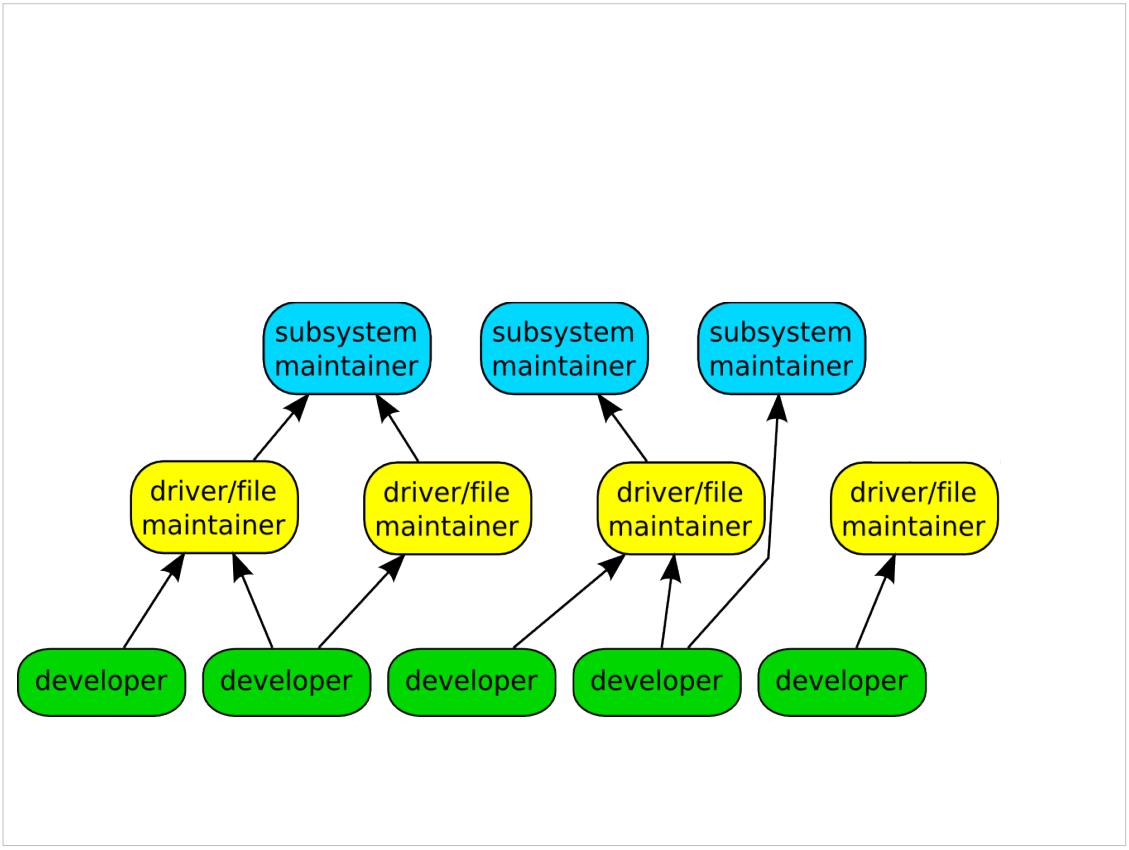
(d) This contribution is public.

This is what “Signed-off-by:” means.
All contributions to the Linux kernel have to agree to this, and every single patch has at least one signed-off-by line, usually all have at least two.

This is also a “blame” trail, showing who changed each line in the kernel, and who agreed with that change.

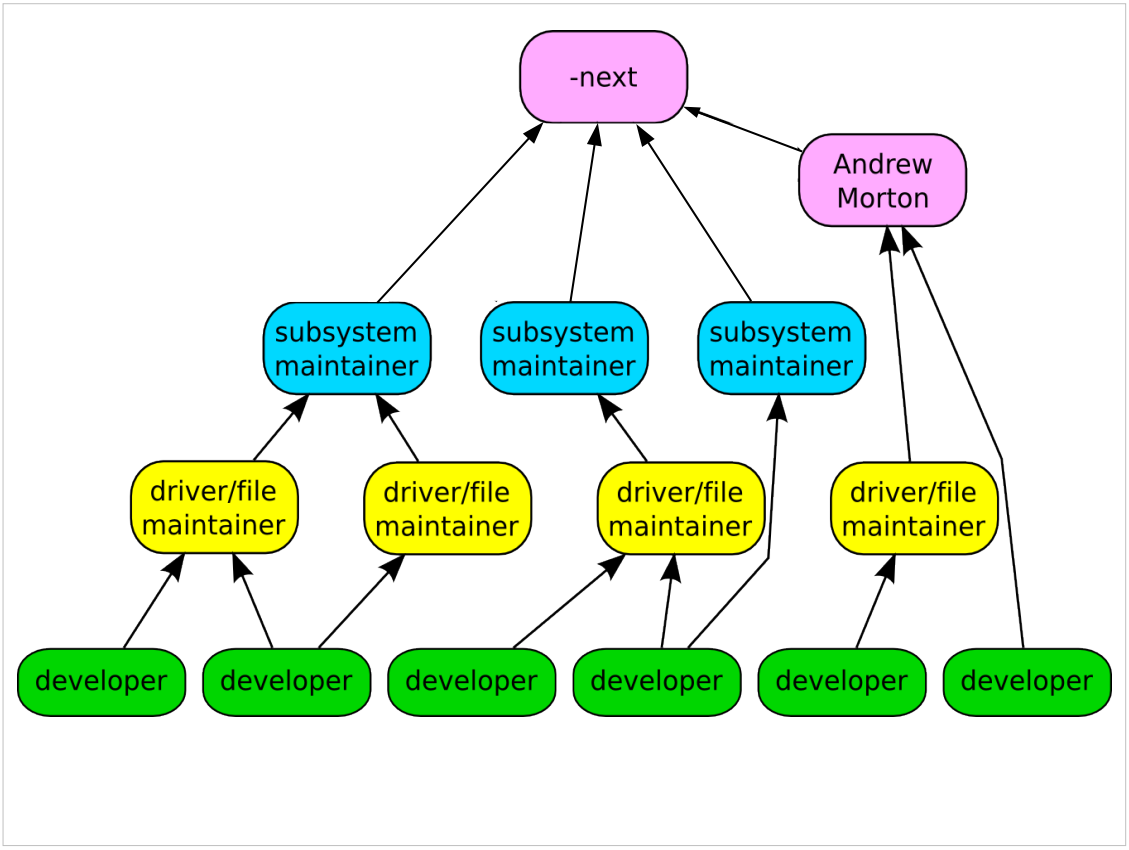
If a problem is found, this is the developers that you can ask about it.

Because of this, every line in the Linux kernel can be traced back to at least two developers who are responsible for it.
This is better than any other body of code.



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.

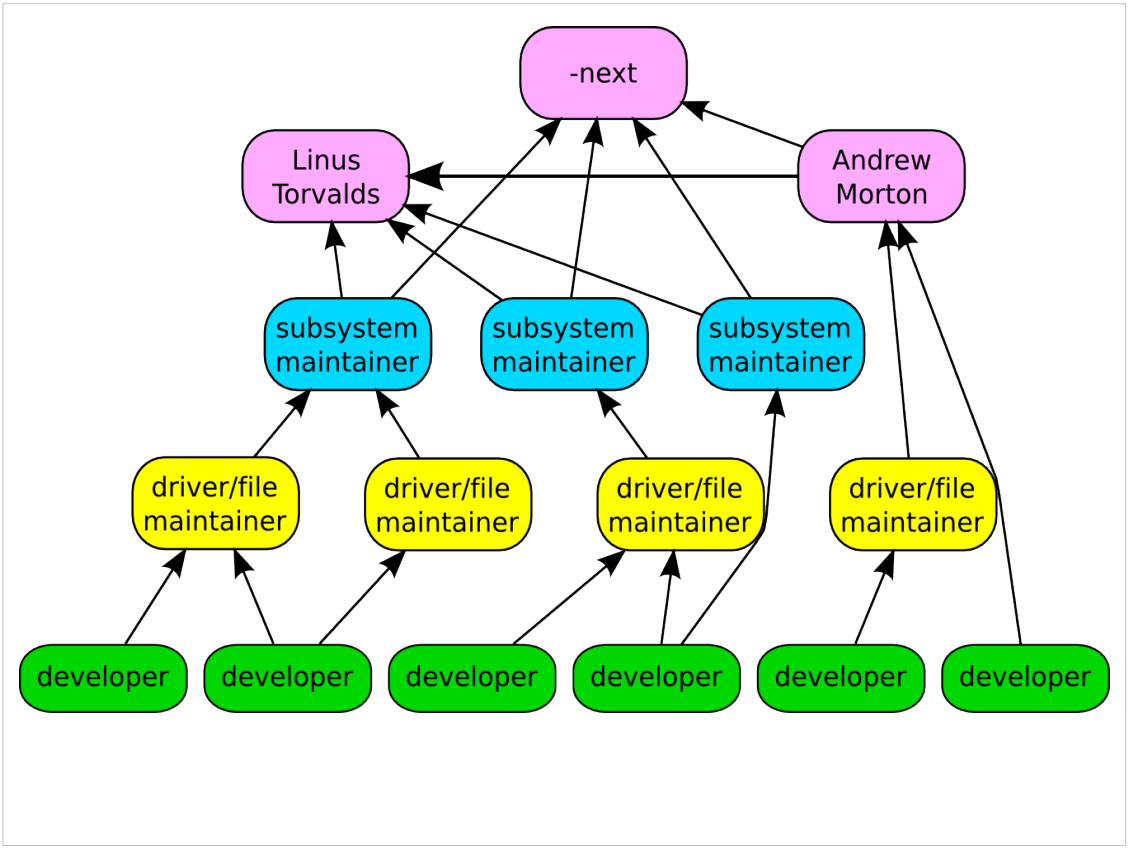
We have around 150 subsystem maintainers



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.

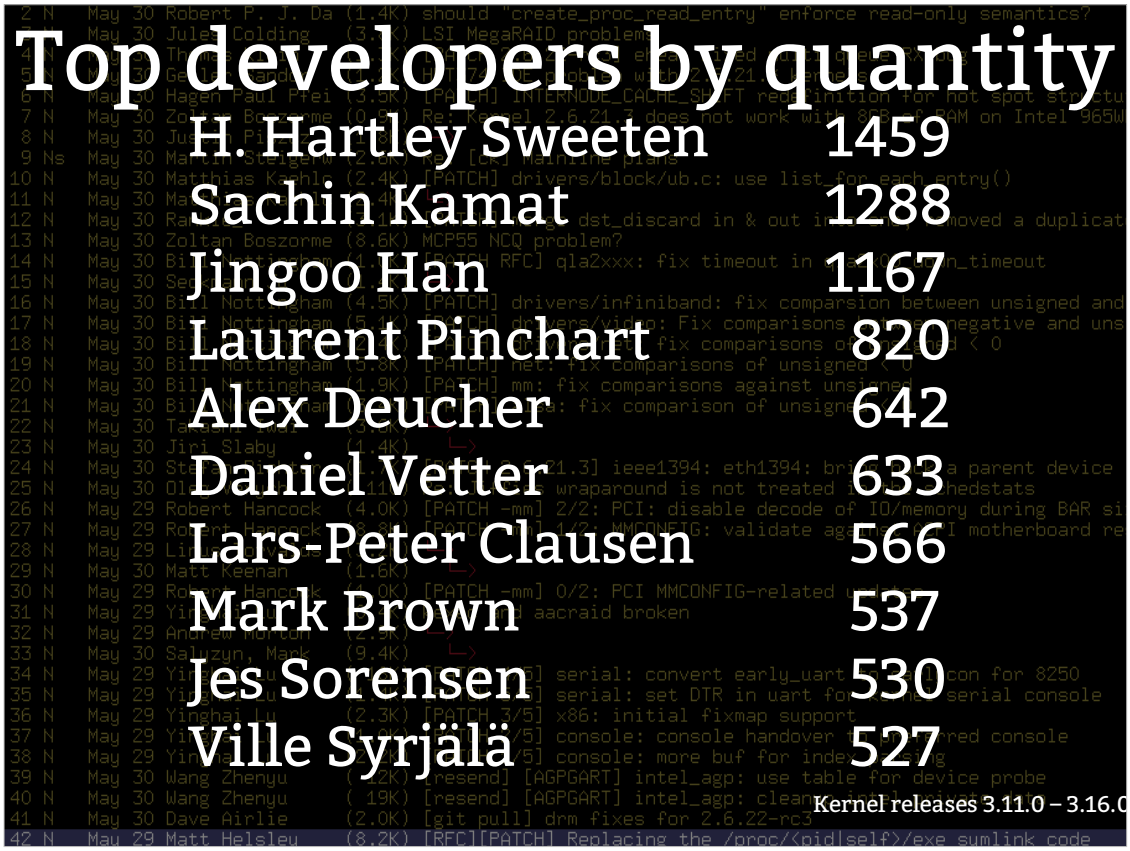


Every 3 months, when the merge window opens up, everything gets sent to Linus from the subsystem maintainers and Andrew Morton.

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.

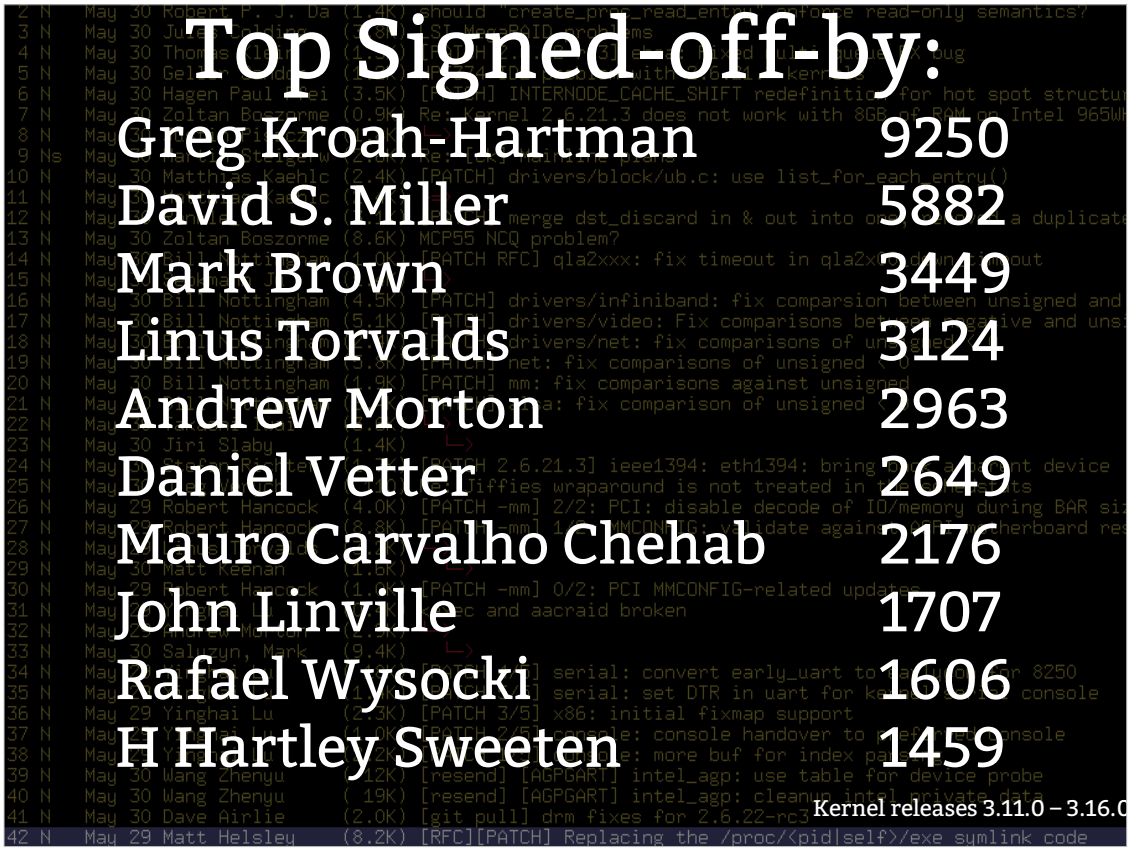
Linux-next can not just be sent to Linus as there are things in there that sometimes are not good enough to be merged just yet, it is up to the individual subsystem maintainer to decide what to merge.



Hartley – comedi
Sachin – exynos ARM platform
Jingoo – backlight / framebuffer
Laurent – video camera drivers
Alex – remote block driver
Daniel – Intel graphics driver
Lars – sound
Mark – embedded sound
Jes – a wireless driver
Ville – Intel graphics driver

Top Signed-off-by:

Greg Kroah-Hartman	9250
David S. Miller	5882
Mark Brown	3449
Linus Torvalds	3124
Andrew Morton	2963
Daniel Vetter	2649
Mauro Carvalho Chehab	2176
John Linville	1707
Rafael Wysocki	1606
H Hartley Sweeten	1459



Greg – driver core, usb, staging
David – networking
Mark – embedded sound
Linus – everything
Andrew – everything
Daniel – Intel graphics
Mauro – v4l
John – wireless networking
Rafael – ACPI / power management
Hartley – comedi data acquisition

So you can view this as either 17% is done by non-affiliated people, or 83% 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.

Vision Engraving (Hartley 1450 patches)

FOSS Outreach Program for Women 1000
20 women interns / students

Amazon 3 patches total.
That sucks.

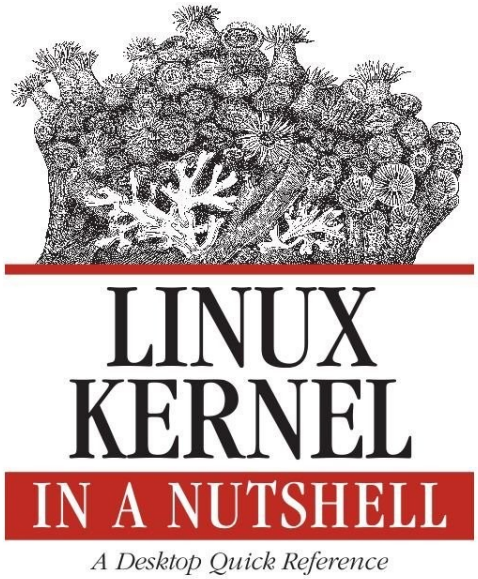
“Working upstream saves time and money”

Dan Frye – VP Open Systems, IBM
Dirk Hohndel – Chief Technologist, Intel

Getting involved

Run the kernel.org release on your machine

Getting involved



This book tells you how to build and install a kernel on your machine.

Free online

Getting involved

`Documentation/HOWTO`

`Documentation/development-process`

These documents in the kernel source directory are the best place to start if you want to understand how the development process works, and how to get involved.

The HOWTO file has links to almost everything else you ever wanted..

Getting involved

kernelnewbies.org

http://www.kernelnewbies.org

Getting involved

Google “write your first kernel patch”

This is a video of a talk I gave at FOSDEM, going through the steps, showing exactly how to create, build, and send a kernel patch.

Getting involved

kernelnewbies.org/KernelJanitors/Todo

So you know how to create a patch, but what should you do? The kernel janitors has a great list of tasks to start with in cleaning up the kernel and making easy patches to be accepted.

Getting involved

Eudyptula Challenge
(little penguin)

http://eudyptula-challenge.org/

Google “Linux kernel challenge” to find the site, if you can't remember Eudyptula.

It is a series of programming challenges, all run through email that starts out with a “Hello World” kernel module, and gets more complex from there. Over 4000 people are currently taking the challenge, and is a lot of fun if you don't know where to start out.

You need knowledge of C, but that's about it.

Getting involved

Linux Driver Project

`drivers/staging/*/TODO`

The staging tree also needs a lot of help, here are lists of things to do in the kernel for the drivers to be able to move out of the staging area.

Please always work off of the linux-next tree if you want to do these tasks, as sometimes they are already done by others by the time you see them in Linus's tree.

Obligatory Penguin Picture