

# Linux Kernel Development

Greg Kroah-Hartman  
gregkh@linux.com

[github.com/gregkh/kernel-development](https://github.com/gregkh/kernel-development)

**37,617 files**  
**14,998,000 lines**

2,823 developers  
379 companies

12,300 lines added

8,800 lines removed

2,300 lines modified

12,300 lines added  
8,800 lines removed  
2,300 lines modified  
  
per day for all of 2011


# 5.77 changes per hour

Kernel releases 2.6.37 – 3.2.0  
January 2011 – January 2012

# How we stay sane

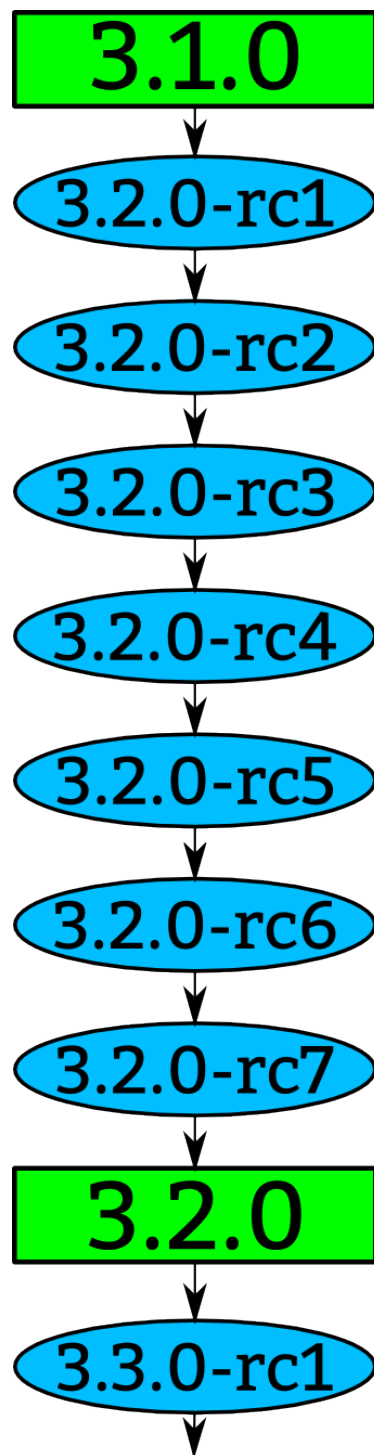
## Time based releases

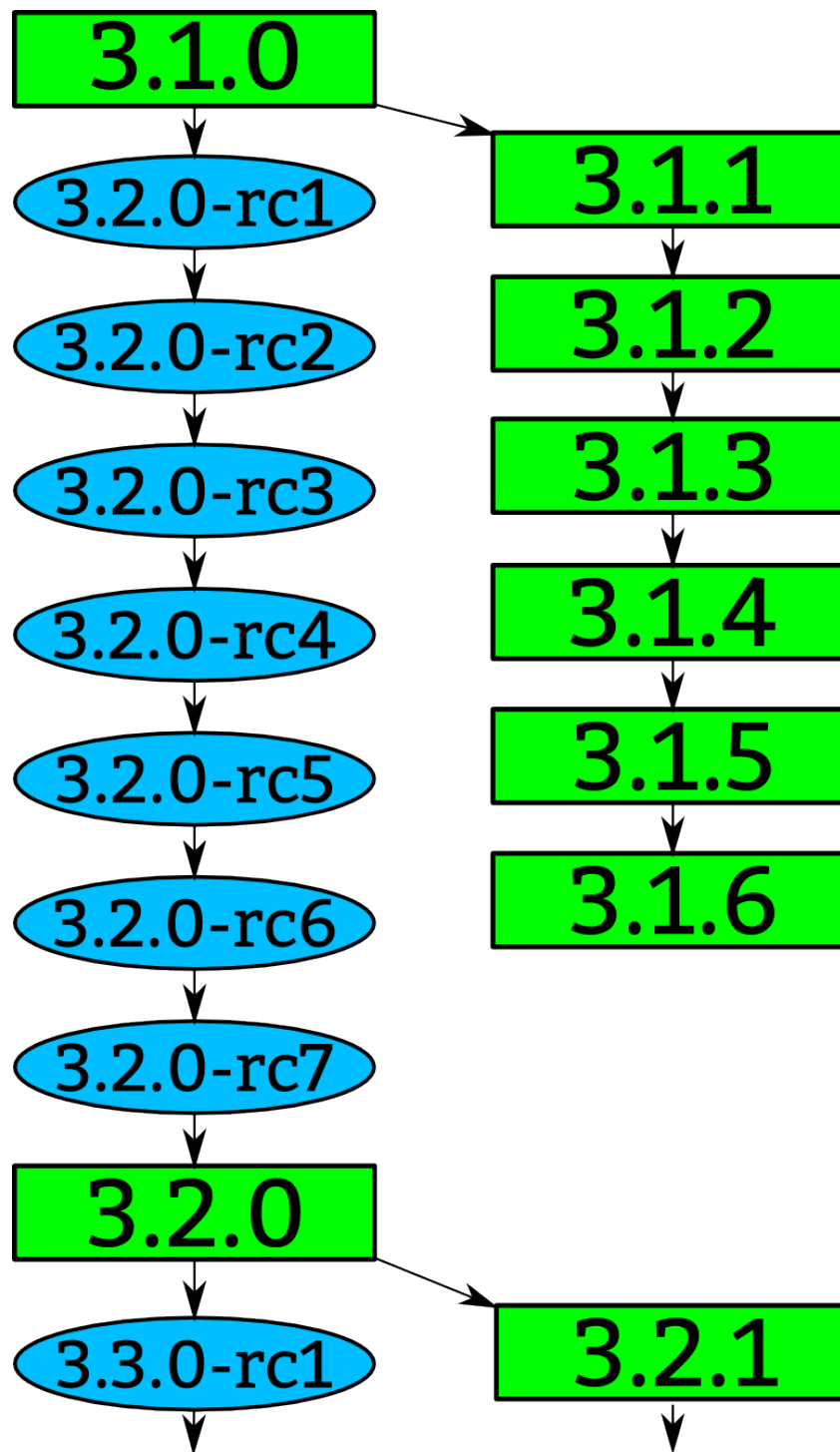
## Incremental changes



**New release every  
2<sup>3</sup>/<sub>4</sub> months**







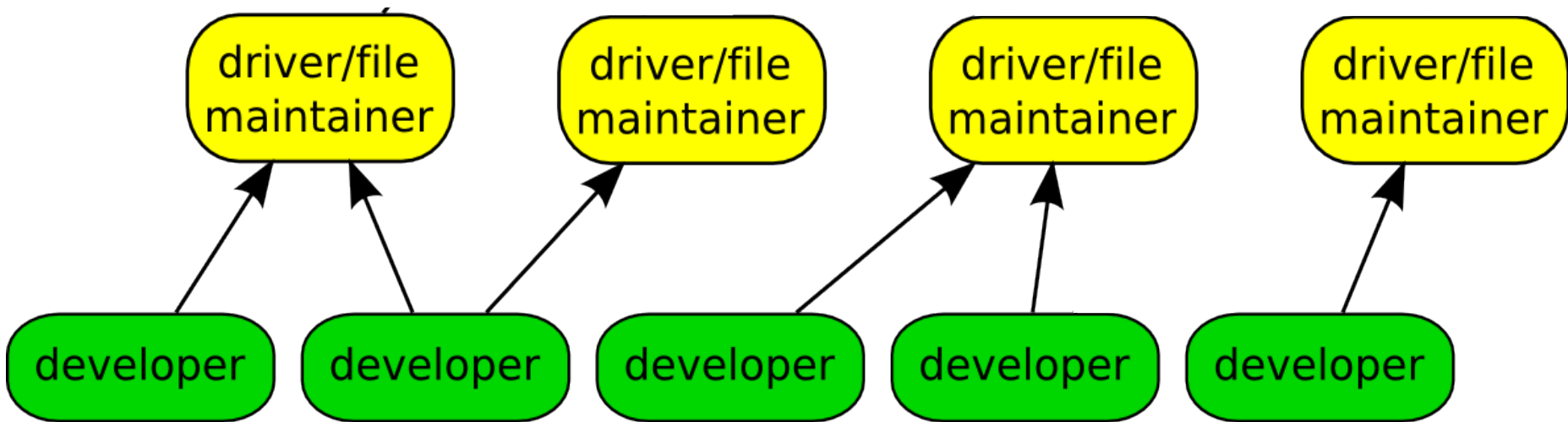
developer

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

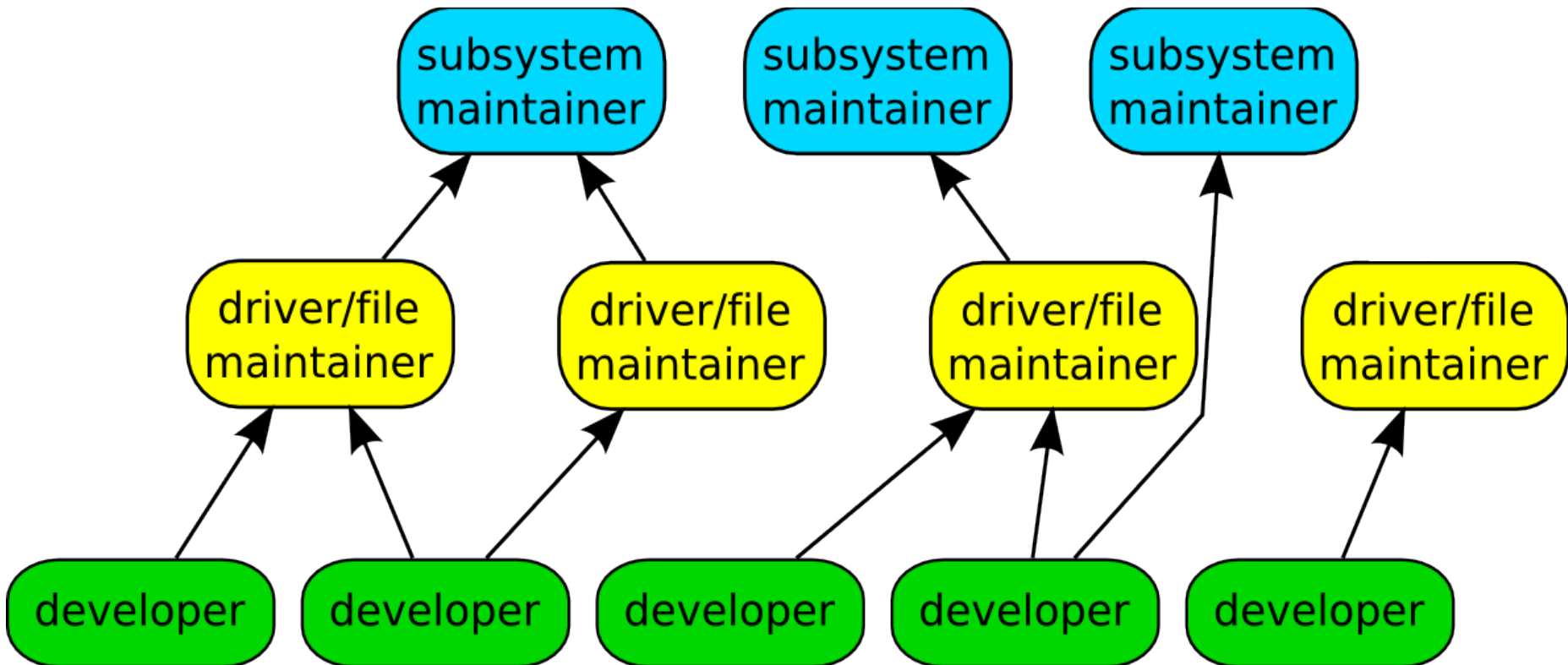
In the case where a gadget driver is removed while no transceiver was found at probe time, a bug in otg\_put\_transceiver() will trigger.

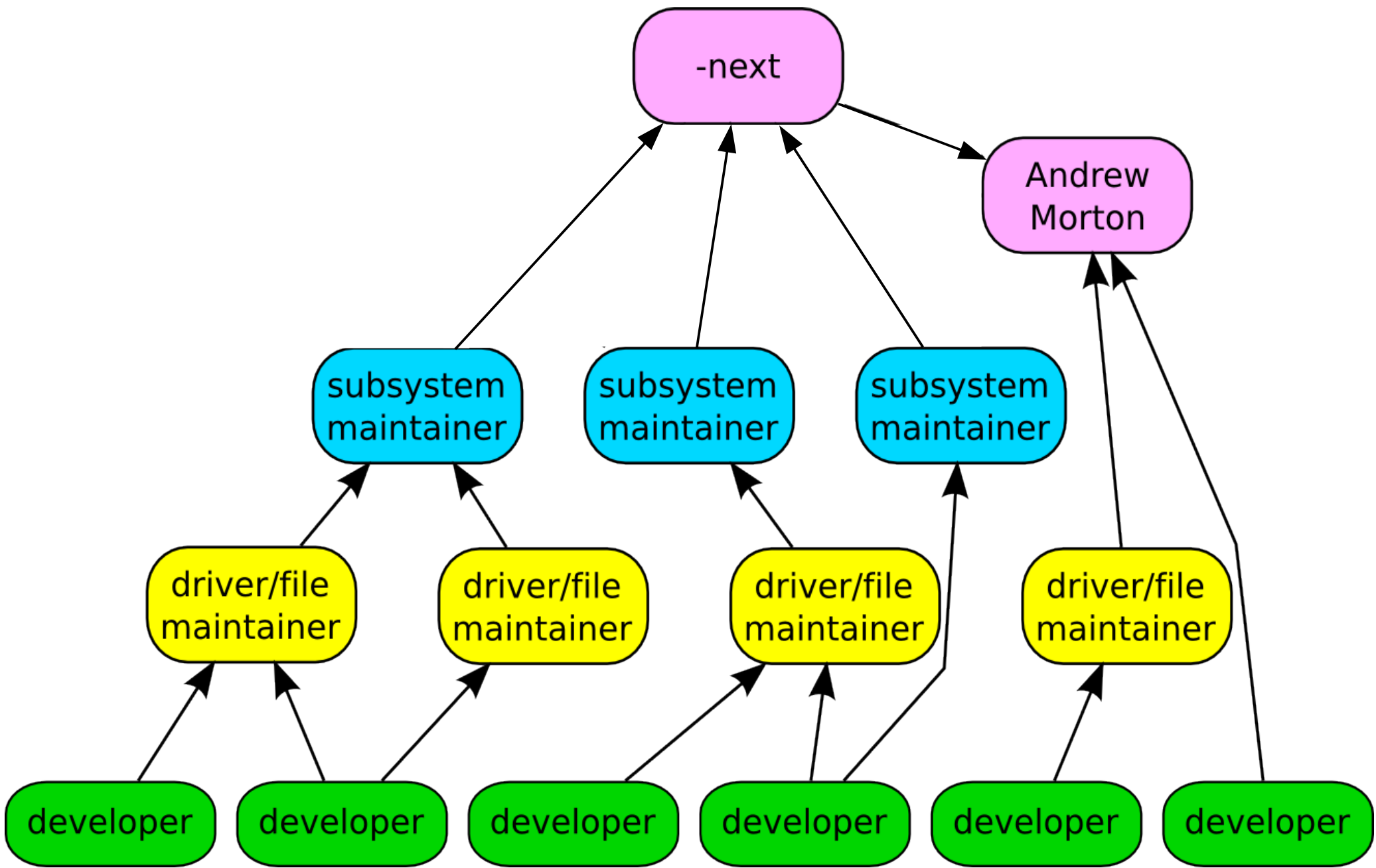
Signed-off-by: Robert Jarzmik <robert.jarzmik@free.fr>  
Acked-by: David Brownell <dbrownell@users.sourceforge.net>  
Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>

```
--- 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);
 }
```

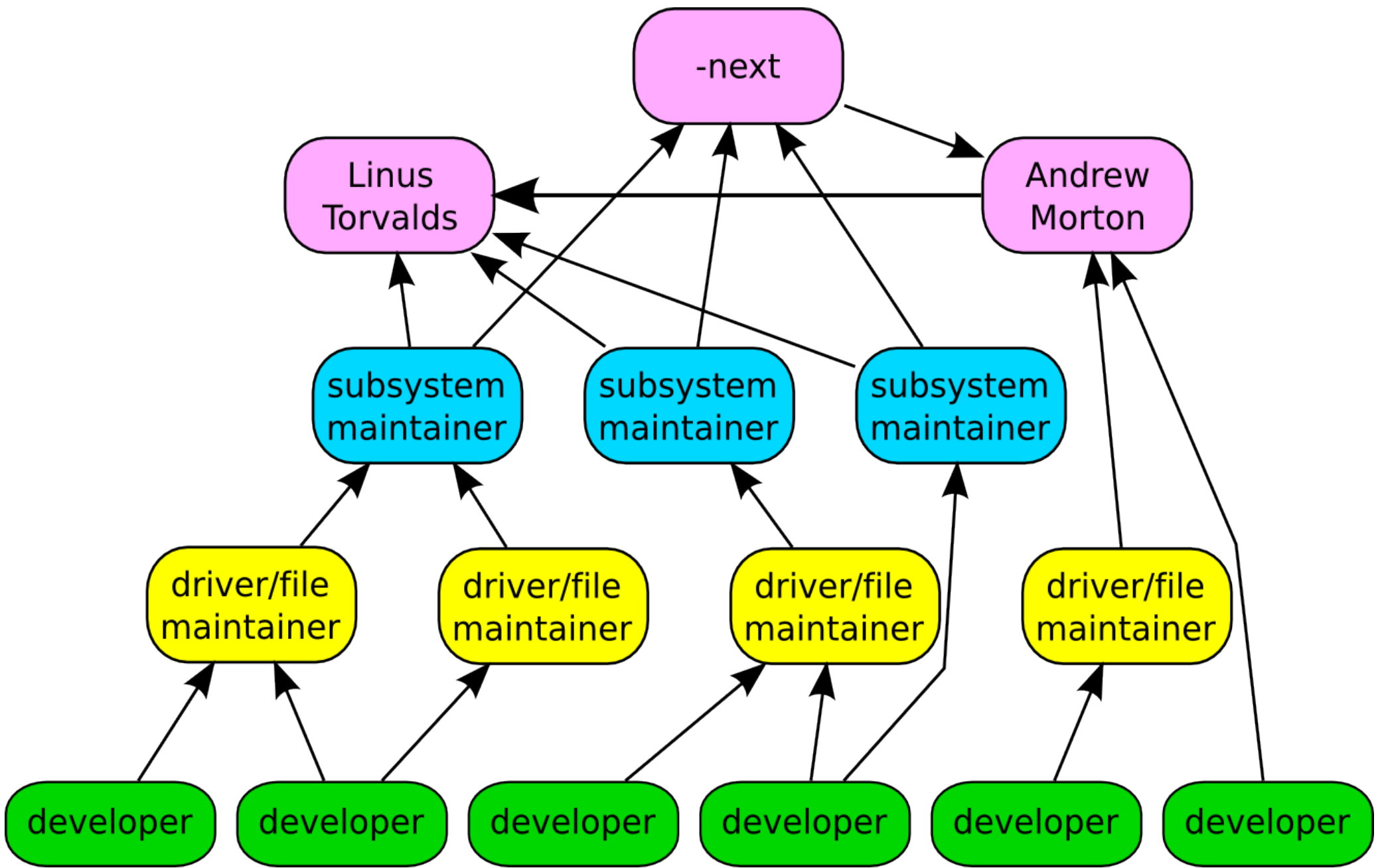
# Developer's Certificate of Origin

- (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.









# Top developers by quantity

Mark Brown	743
Thomas Gleixner	643
K. Y. Srinivasan	588
David Miller	533
Joe Perches	493
Axel Lin	444
Al Viro	426
Russell King	426
Takashi Iwai	409
Ben Skeggs	408

# Top Signed-off-by:

Greg Kroah-Hartman 5807

David S. Miller 4047

John Linville 3024

Mauro Carvalho Chehab 2237

Linus Torvalds 2120

Mark Brown 1704

Andrew Morton 1642

James Bottomley 1110

Takashi Iwai 957

Russell King 934

Kernel releases 2.6.37 – 3.2.0

# Who is funding this work?

- 1.
2. Red Hat 10.3%
3. Intel 7.3%
- 4.
5. Novell 4.0%
6. IBM 3.6%
7. Texas Instruments 3.4%
8. Broadcom 2.9%
9. Consultants 2.4%
10. Nokia 1.8%

# Who is funding this work?



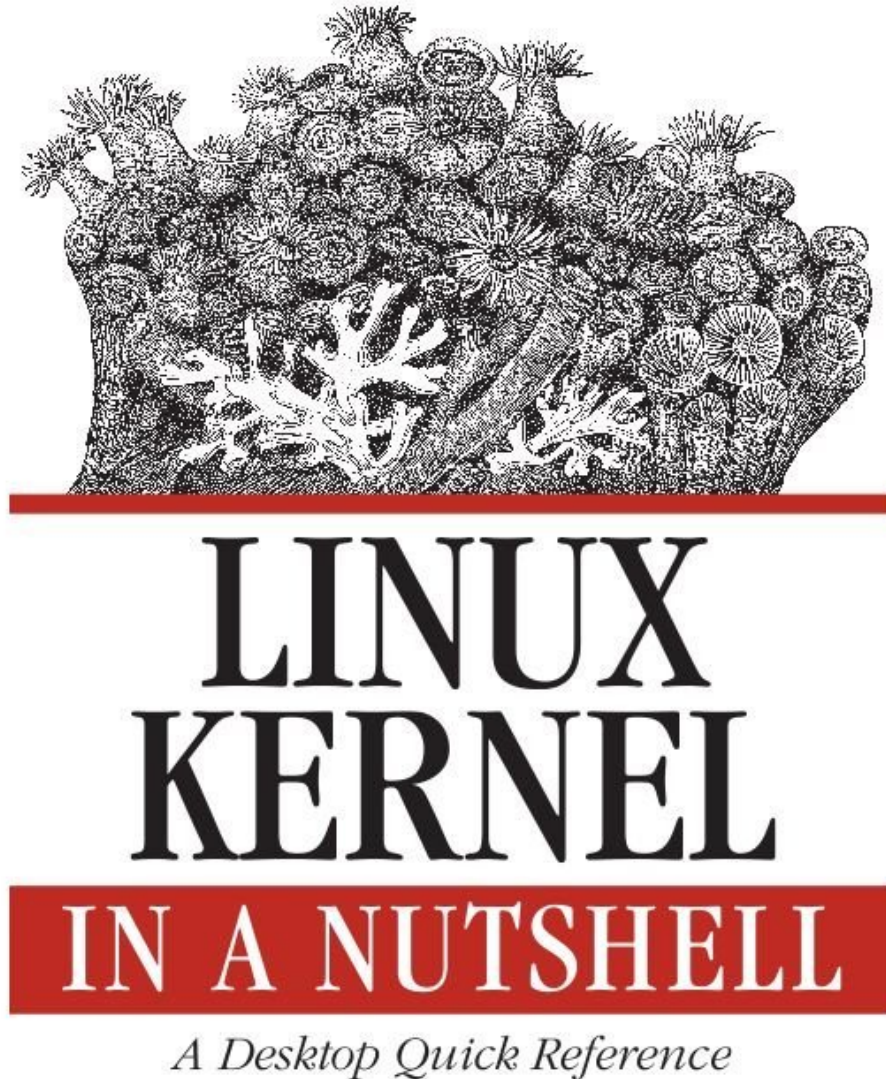
1. “Amateurs”	15.9%
2. Red Hat	10.3%
3. Intel	7.3%
4. Unknown Individuals	4.4%
5. Novell	4.0%
6. IBM	3.6%
7. Texas Instruments	3.4%
8. Broadcom	2.9%
9. Consultants	2.4%
10. Nokia	1.8%

# Getting involved

Run the kernel.org release on your machine



# Getting involved



# Getting involved

**Documentation/HOWTO**

**Documentation/development-process**



# Getting involved

[kernelnewbies.org](http://kernelnewbies.org)



# Getting involved

Google “write your first kernel patch”

# Getting involved

[kernelnewbies.org/KernelJanitors/ToDo](https://kernelnewbies.org/KernelJanitors/ToDo)

# Getting involved

Linux Driver Project

`drivers/staging/*/TODO`



[github.com/gregkh/kernel-development](https://github.com/gregkh/kernel-development)



# Linux Kernel Development

Greg Kroah-Hartman  
gregkh@linux.com

[github.com/gregkh/kernel-development](https://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.

**37,617 files**  
**14,998,000 lines**

Kernel release 3.2

This was for the 3.2 kernel release, which happened January 4, 2012.



# 2,823 developers 379 companies

Kernel releases 2.6.37 – 3.2.0  
January 2011 – January 2012

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.

**12,300 lines added**  
**8,800 lines removed**  
**2,300 lines modified**

Kernel releases 2.6.37 – 3.2.0  
January 2011 – January 2012

**12,300 lines added  
8,800 lines removed  
2,300 lines modified  
  
per day for all of 2011**

Kernel releases 2.6.37 – 3.2.0  
January 2011 – January 2012

# 5.77 changes per hour

Kernel releases 2.6.37 – 3.2.0  
January 2011 – January 2012

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.

# How we stay sane

## Time based releases

## Incremental changes

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

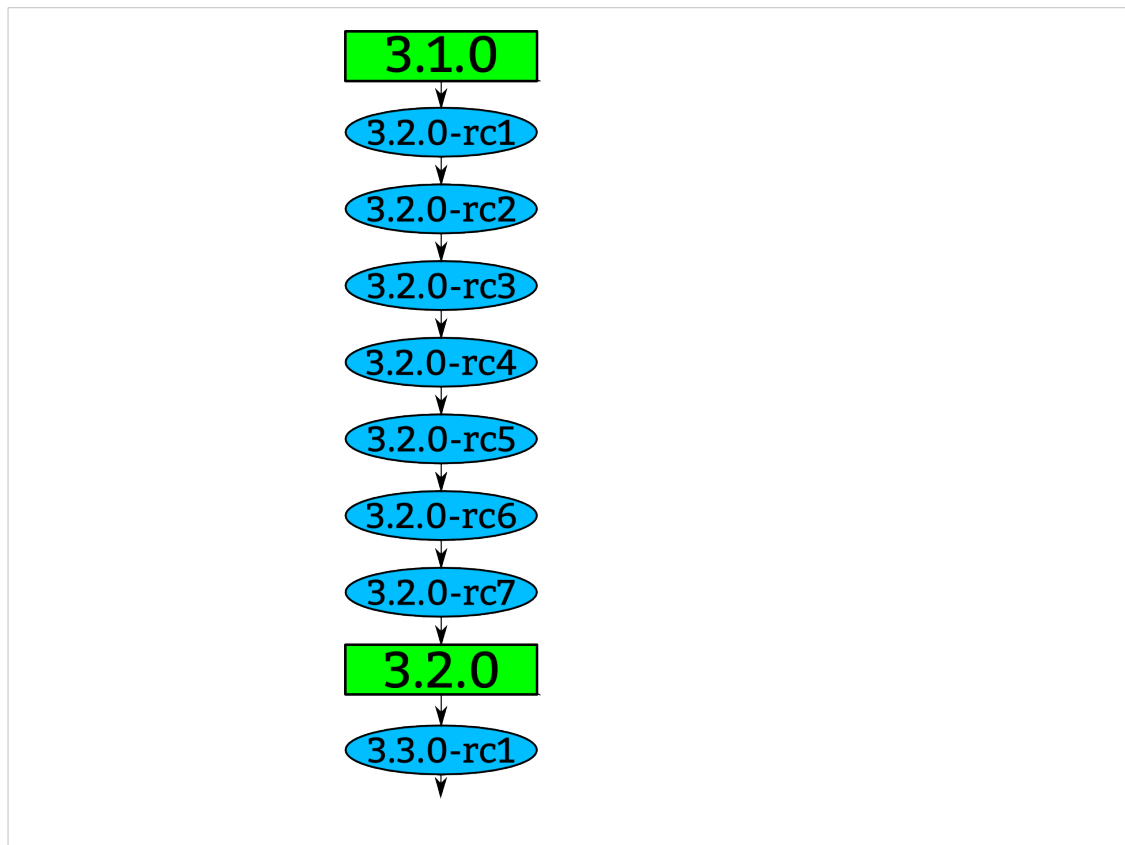
We went this fast the year before this as well, this is an amazing rate of change.

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.



84 days to be exact, very regular experience.

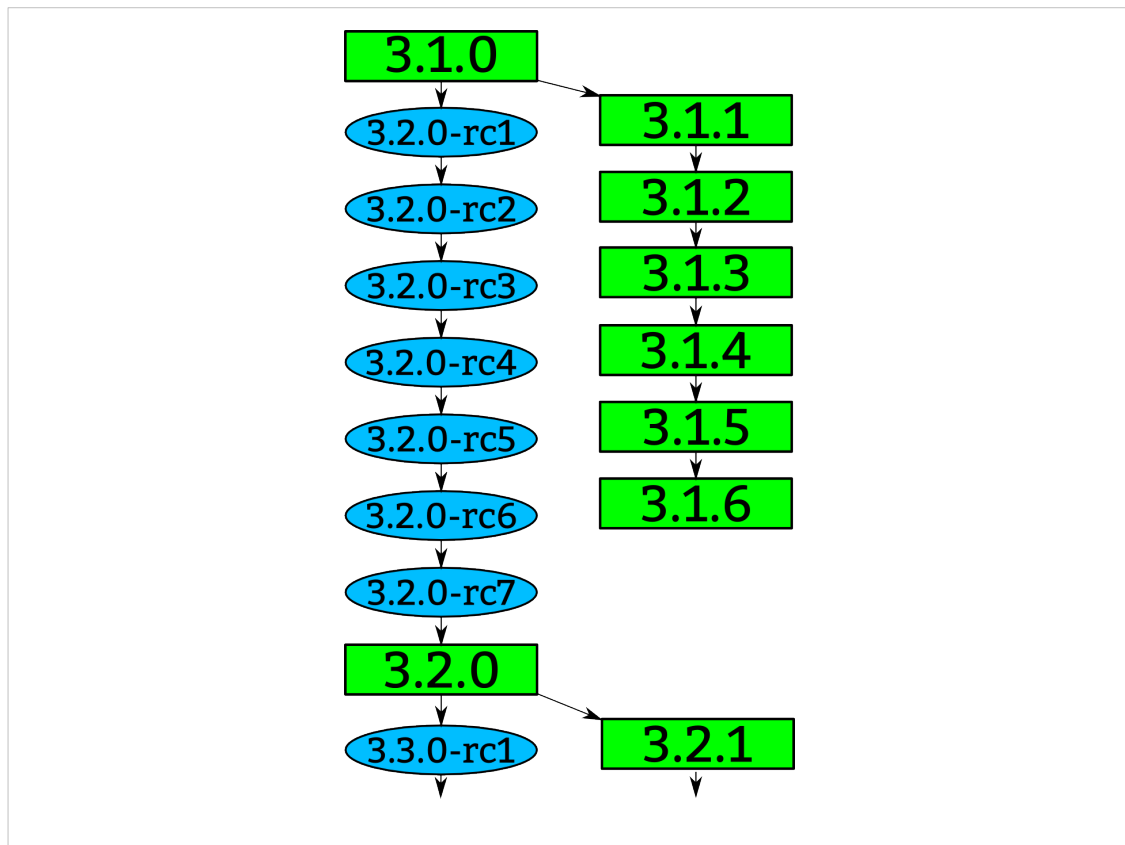


How a kernel is developed.

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.

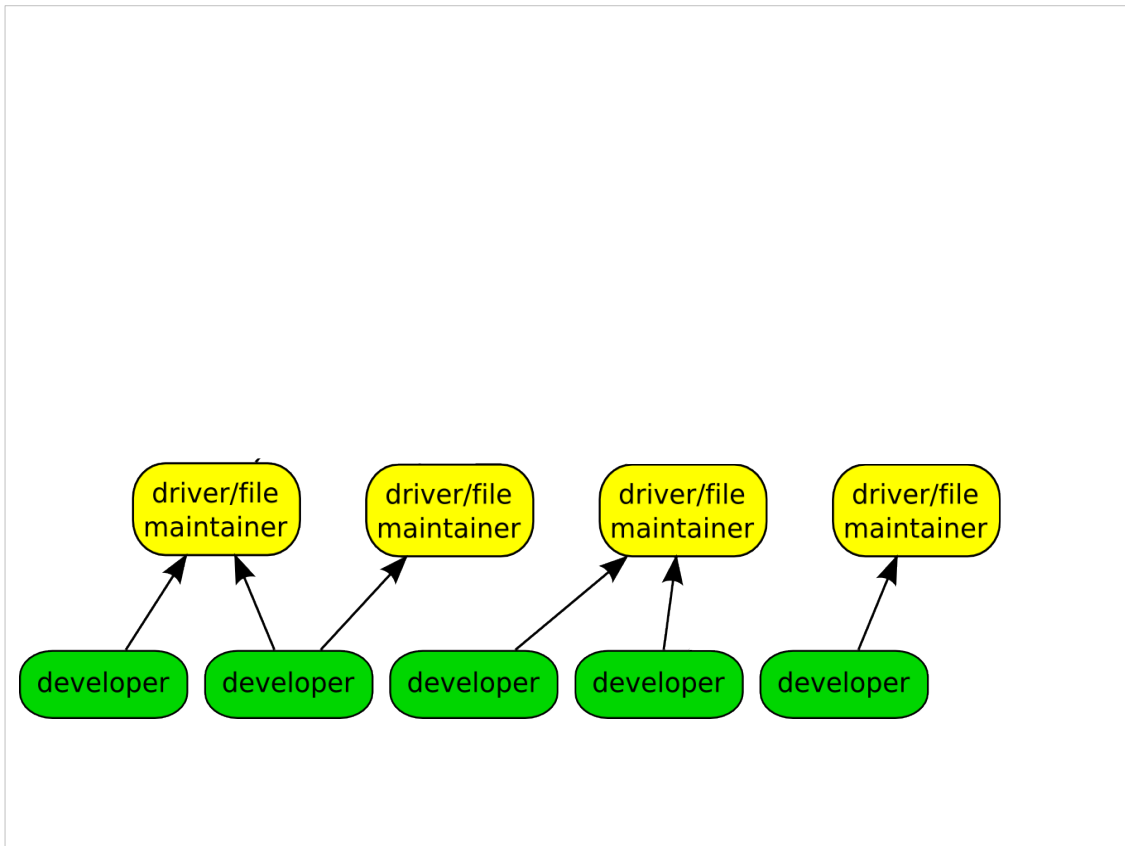
With the exception of “longterm” stable releases, those are special, they stick around for much longer...





Like mentioned before, we have almost 2900 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>
    Aacked-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 me before it was committed 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.

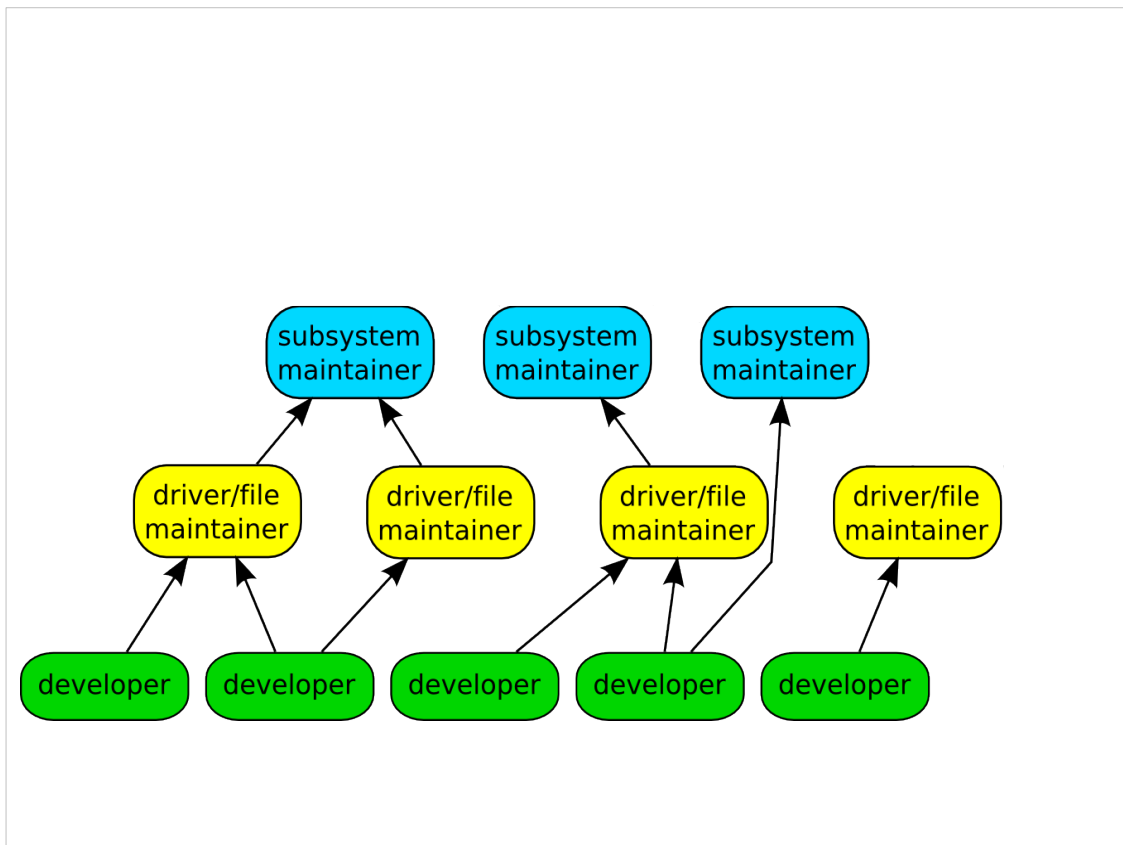
This is what “Signed-off-by:” means.

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.

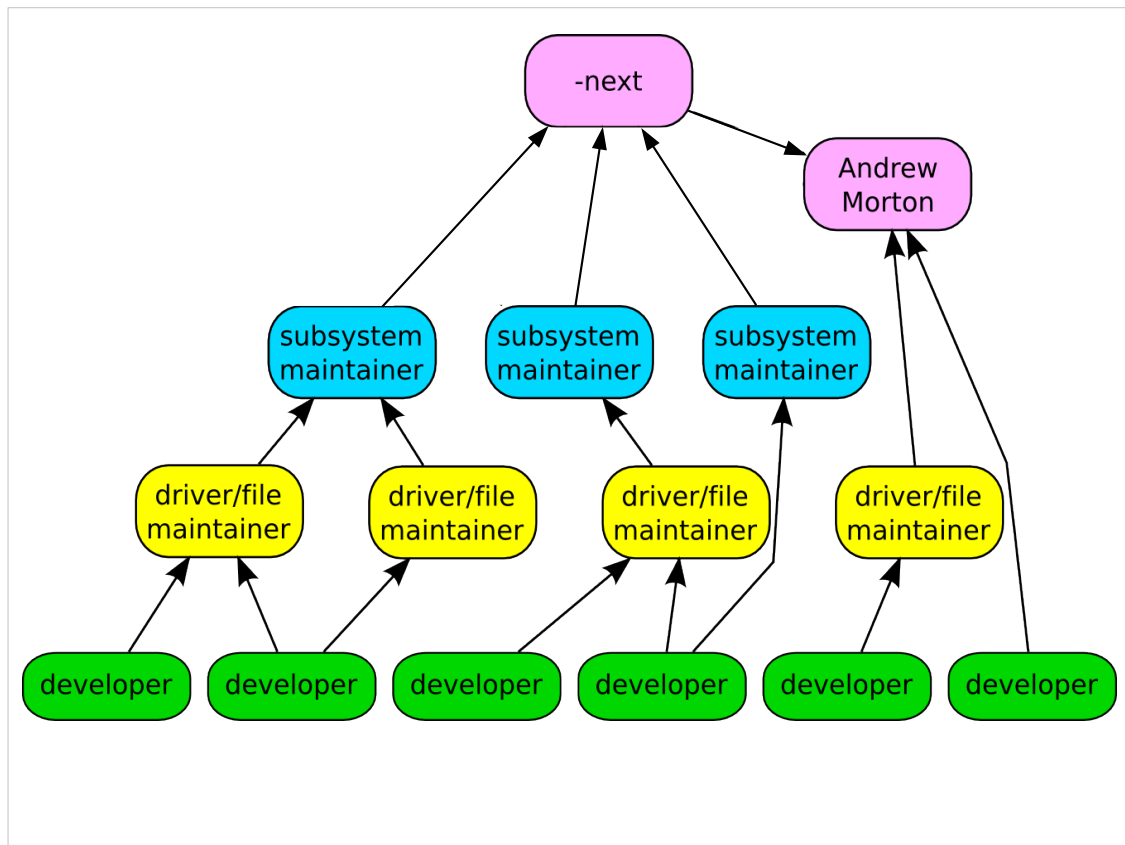
If a problem is found, this is the developers that you can ask about it.

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.

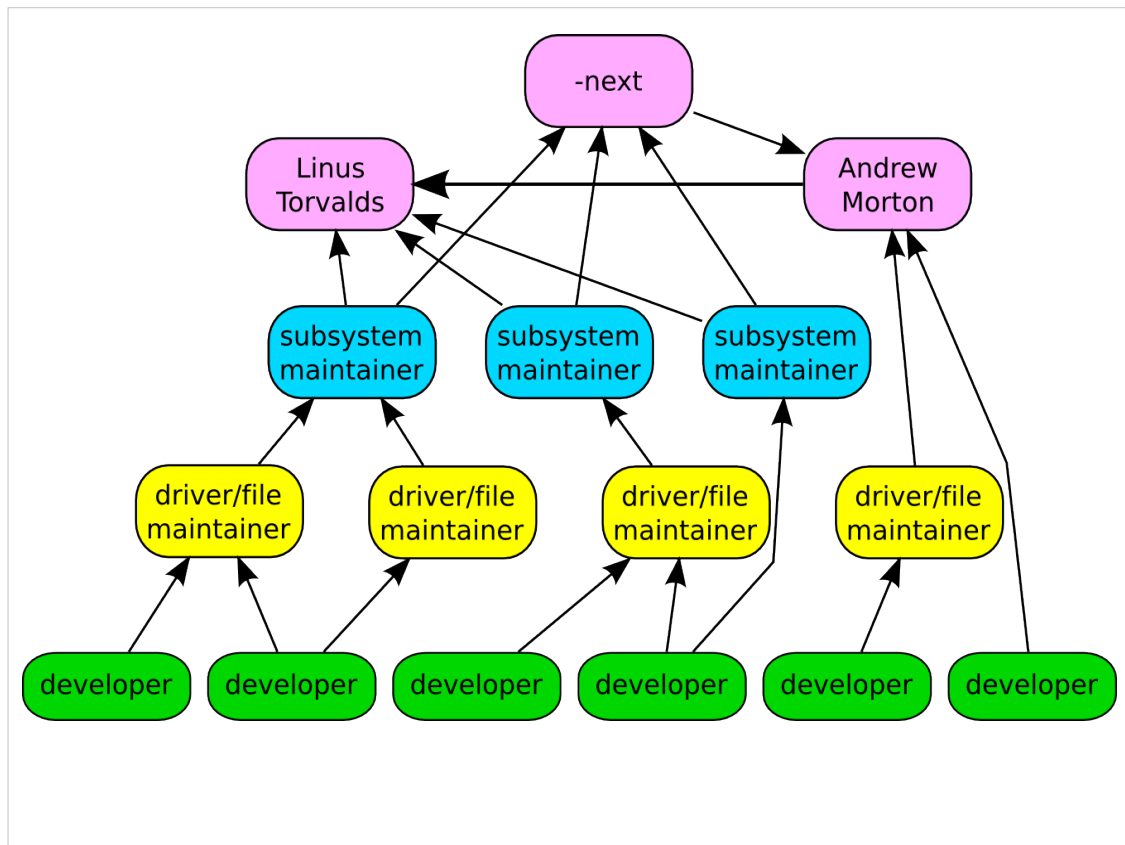
We have around 150 subsystem maintainers



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.

# Top developers by quantity

6 N	May 30	Robert P. J. Da	(1.4K)	[PATCH] "create_proc_read_entry" enforce read-only semantics?	743
7 N	May 30	Thomas Gleixner	(2.4K)	[PATCH] drivers/block/ub.c: use list_for_each_entry()	643
8 N	May 30	K. Y. Srinivasan	(1.0K)	[PATCH RFC] glazxxx: fix timeout in glaz200_down_timeout	588
9 Ns	May 30	David Miller	(5.1K)	[PATCH] drivers/net: fix comparisons of unsigned < 0	533
10 N	May 30	Joe Perches	(1.9K)	[PATCH] mm: fix comparisons against unsigned	493
11 N	May 30	Axel Lin	(1.4K)	[PATCH] ieee1394: bring back a parent device	444
12 N	May 30	Al Viro	(8.8K)	[PATCH -mm] 1/2: MMCONFIF: validate against ACPI motherboard re	426
13 N	May 30	Russell King	(9.4K)	[PATCH -mm] 0/2: PCI MMCONFIF-related updates	426
14 N	May 30	Takashi Iwai	(2.3K)	[PATCH 2/5] console: console handover to preferred console	409
15 N	May 30	Ben Skeggs	(1.0K)	[PATCH 1/5] console: more buf for console using	408
16 N	May 30		(1.9K)	[resend] [AGPGART] intel_agp: use local device probe	
17 N	May 30		(2.0K)	[git pull] drm fixes for 2.6.22-rc	
18 N	May 29	Matt Helsley	(8.2K)	[RFC][PATCH] Replacing the /proc/<pid/self>/exe symlink code	

Kernel releases 2.6.37 – 3.2.0

Mark – embedded sound

Thomas – x86 maintainer

KY – hyperv

David – networking

Joe – janitorial

Alexl – janitorial

Al – vfs and filesystem

Russell – ARM maintainer

Takashi – sound maintainer

Ben – nouveau developer



Top Signed-off-by:		
Greg Kroah-Hartman	5807	
David S. Miller	4047	
John Linville	3024	
Mauro Carvalho Chehab	2237	
Linus Torvalds	2120	
Mark Brown	1704	
Andrew Morton	1642	
James Bottomley	1110	
Takashi Iwai	957	
Russell King	934	
Kernel releases 2.6.37 – 3.2.0		

Greg – driver core, usb, staging

David – networking

John – wireless networking

Mauro - v4l

Linus – everything

Mark - embedded

Andrew – everything

James – SCSI

Takashi – sound

Russell - ARM

# Who is funding this work?

1.	
2. Red Hat	10.3%
3. Intel	7.3%
4.	
5. Novell	4.0%
6. IBM	3.6%
7. Texas Instruments	3.4%
8. Broadcom	2.9%
9. Consultants	2.4%
10. Nokia	1.8%

Kernel releases 2.6.37 – 3.2.0

11 Wolfson Microelectronics	830 (1.7%)
12 Google	794 (1.6%)
13 Samsung	778 (1.6%)
14 Oracle	728 (1.5%)
15 Microsoft	672 (1.4%)
16 AMD	669 (1.3%)
17 Freescale	625 (1.3%)
18 tglx PITA	591 (1.2%)
19 Fujitsu	564 (1.1%)
20 Atheros Communications	543 (1.1%)
21 (Academia)	534 (1.1%)
22 Wind River	513 (1.0%)
23 Linaro	503 (1.0%)
24 ST Ericsson	457 (0.9%)
25 MiTAC	444 (0.9%)
26 Pengutronix	428 (0.9%)
27 Analog Devices	418 (0.8%)
28 Marvell	383 (0.8%)
29 University of Cambridge	360 (0.7%)
30 Societe Francaise de Radiotelephone	334 (0.7%)

# Who is funding this work?

1. "Amateurs"	15.9%
2. Red Hat	10.3%
3. Intel	7.3%
4. Unknown Individuals	4.4%
5. Novell	4.0%
6. IBM	3.6%
7. Texas Instruments	3.4%
8. Broadcom	2.9%
9. Consultants	2.4%
10. Nokia	1.8%

Kernel releases 2.6.37 – 3.2.0

So you can view this as either 20% is done by non-affiliated people, or 80% is done by companies.

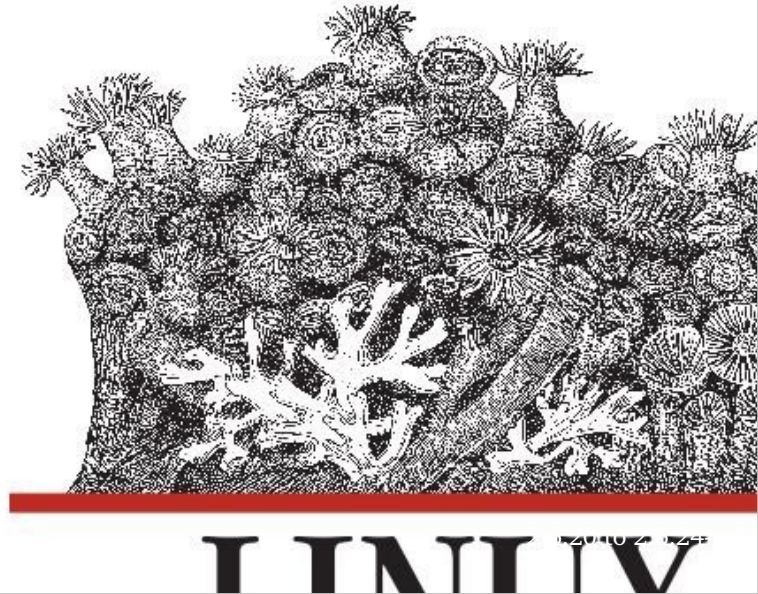
Now to be fair, if you show any skill in kernel development you are instantly hired.

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.

# 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](https://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

## 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

