I don't want your code

Linux Kernel Maintainers,
why are they so grumpy?

Greg Kroah-Hartman
gregkh@linuxfoundation.org

2,829 developers
407 companies

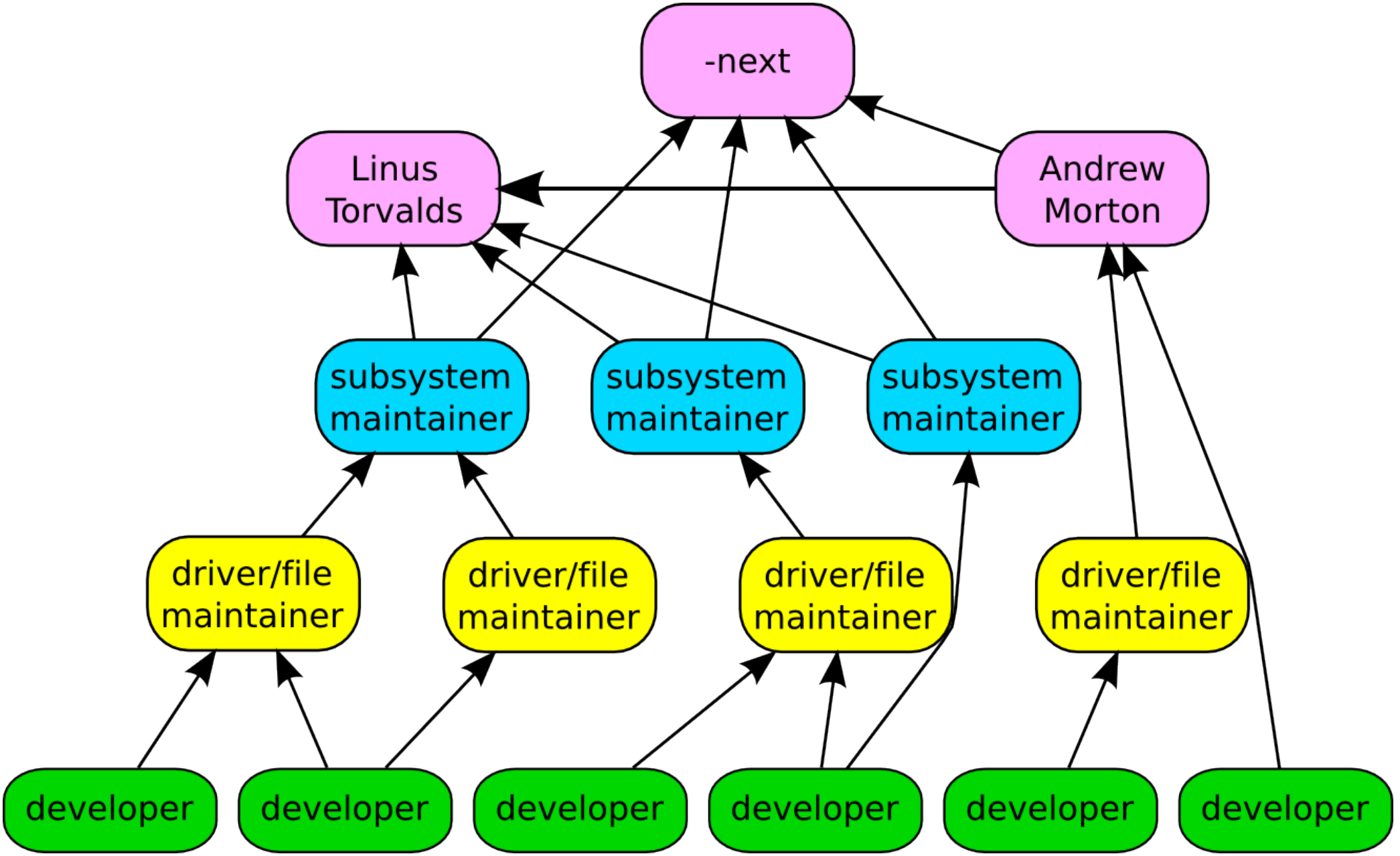
Kernel releases 3.4.0 – 3.8.0
March 2012 – Feb 2013

6.98 changes per hour

Kernel releases 3.4.0 – 3.8.0
March 2012 – Feb 2013

7.38 changes per hour

3.8.0 release


-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	1438
Mark Brown	642
Al Viro	553
Axel Lin	532
Greg Kroah-Hartman	505
Daniel Vetter	418
Johannes Berg	403
Bill Pemberton	394
David Miller	387
Sachin Kamet	387

Kernel releases 3.4.0 – 3.8.0

Top Signed-off-by:

Greg Kroah-Hartman	7004
David S. Miller	3883
Mark Brown	2450
Mauro Carvalho Chehab	2436
John Linville	2213
Linus Torvalds	2193
Andrew Morton	1960
H. Hartley Sweeten	1450
Daniel Vetter	1044
Al Viro	981

Kernel releases 3.4.0 – 3.8.0

Who is funding this work?

1. “Amateurs”	13.9%
2. Red Hat	10.1%
3. Intel	8.4%
4. Unknown Individuals	4.5%
5. Linaro	4.4%
6. Texas Instruments	4.0%
7. Vision Engraving	3.2%
8. Novell	3.1%
9. IBM	3.0%
10. Google	2.3%

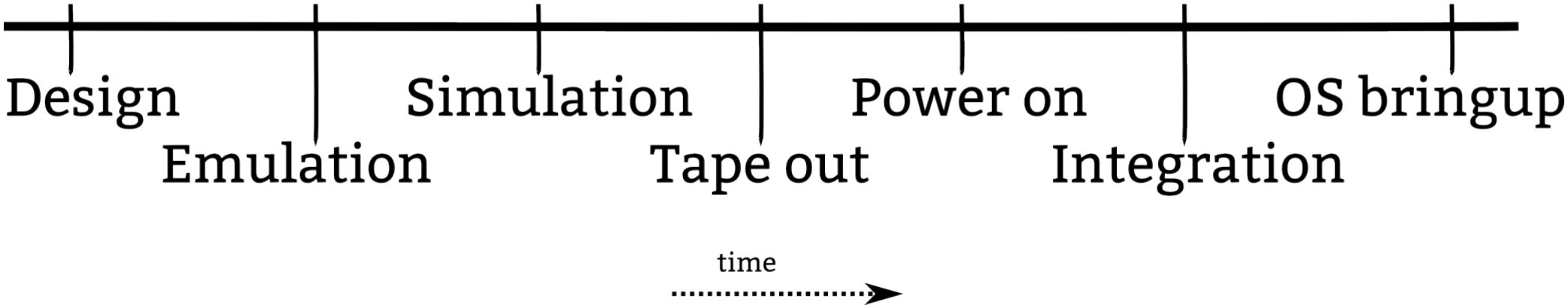
Kernel releases 3.4.0 – 3.8.0

Who is funding this work?

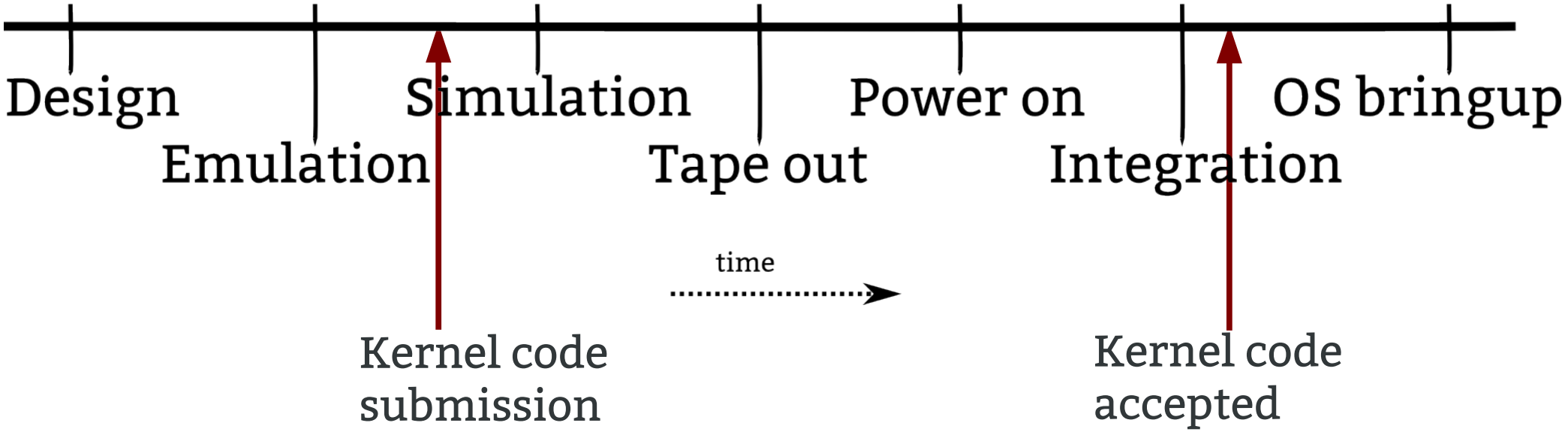
11. Samsung	2.2%
12. Wolfson Micro	1.5%
13. LINBIT	1.5%
14. Consultants	1.4%
15. Linux Foundation	1.3%
16. Nvidia	1.2%
17. Oracle	1.2%
18. Freescale	1.2%
19. Ingics Technology	1.2%
20. Broadcom	1.0%

Kernel releases 3.4.0 – 3.8.0

Development Process



Development Process



“Working upstream saves time and money”

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

Maintainers are like editors in the publishing industry.

– David Miller

Patches I received in a 2 week period

Patches I received in a 2 week period

487

Subject: [PATCH 48/48] ...

15 patch series, no order given

Patches 1, 3-10

“Signed-off-by:” in signature

Signature saying email was confidential

Tabs were converted to spaces

Leading spaces removed

diff in non-unified format

Patch created in driver directory

Patch created in /usr/src/linux-2.6.32

Made against different tree

Wrong coding style

Wrong coding style, and acknowledged it

Would not compile

Broke the build on patch 3/6

Broke the build on patch 3/6 and fixed it on 6/6

Broke the build on patch 5/8

Broke the build on patch 5/8

Contained note that fix would be sent later

Patches that had nothing to do with me

1 patch, 450kb big (4500 lines added)

Obviously wrong kerneldoc

This was a calm two weeks

Case study of a Linaro patch submission

8 patch series for USB

No description of why they were needed

Half of the patches broke the build

Patches resent,
with no description of what changed

Replacments sent again, individually, out of order

Third resend, no ordering at all.

Fourth resend,
described as second version

Insisted that 2 of these HAD to go into 3.9-rc1.

Linaro senior developer resent just the two patches.

Those two required a follow-on patch to fix the Kconfig descriptions

Patches broke all non-DT systems

Driver maintainer returned from vacation and reviewed them

Driver maintainer returned from vacation and reviewed them

Every single one was broken

Driver maintainer returned from vacation and reviewed them

Every single one was broken

“...you can answer at lot of questions like this for yourself very easily, simply by reading the source code.”

I will not accept a patch directly from this developer, for a very long time

```
static void tty_slave_release(struct device *dev)
{
  struct tty_slave *tts = to_tty_slave(dev);

  kfree(tts);
  /* Test code to see if slave device get released */
  BUG();
}
```

It is in my self-interest to ignore your patch

Give me no excuse to reject your patch

What I will do for you:

Review your patch within 1-2 weeks

Offer semi-constructive criticism

Let you know the status of your patch

“Publicly making fun of people is half the fun of open source programming.

In fact the main reason to eschew programming in closed environments is that you can't embarrass people in public.”

– Linus Torvalds

“Publicly making fun of people is half the fun of open source programming.

In fact the main reason to eschew programming in closed environments is that you can't embarrass people in public.”

– Linus Torvalds

github.com/gregkh/presentation-maintainer

I don't want your code

Linux Kernel Maintainers,
why are they so grumpy?

Greg Kroah-Hartman
gregkh@linuxfoundation.org

2,829 developers
407 companies

Kernel releases 3.4.0 – 3.8.0
March 2012 – Feb 2013

This makes the Linux kernel the largest contributed body of software out there that has been created..

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.

6.98 changes per hour

Kernel releases 3.4.0 – 3.8.0
March 2012 – Feb 2013

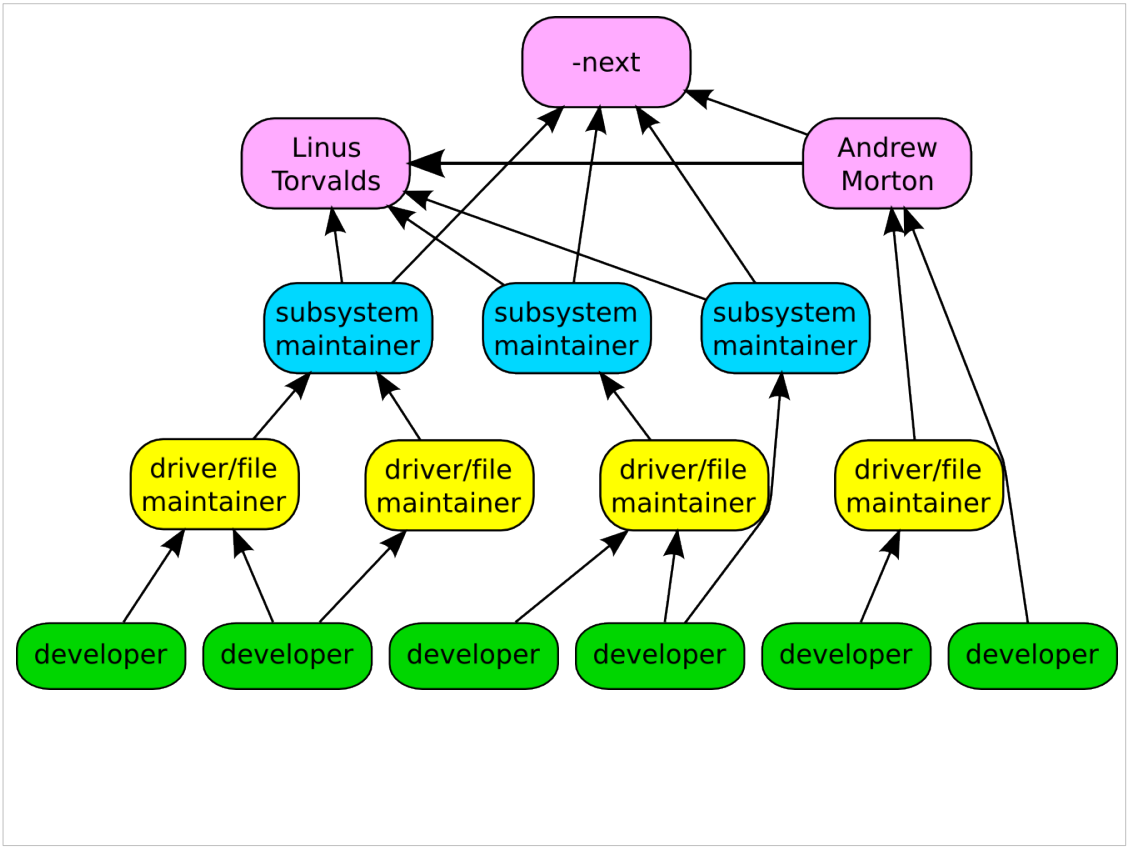
For that year of development, we went at this rate, 24 hours a day, 7 days a week. This is up from last year, which was at 5.2 or so, so we are increasing, which is scary, right?

7.38 changes per hour

3.8.0 release

This past 3.8 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.



Here's a picture of our development model, in a simplified form.

We have about 3000 different developers. They make a patch, and send it through email to the file/driver maintainer. We have about 700 different maintainers listed in the kernel tree at the moment. That maintainer reviews it, and if they accept it, they forward it on to the subsystem maintainer. We have around 85 different subsystem maintainers at the moment, and there are about 160 different subsystem trees that get merged to Linus.

Those maintainers have public kernel trees that all get merged into the linux-next release every day. Then, when the merge window opens up, the subsystem maintainers send their stuff to Linus.

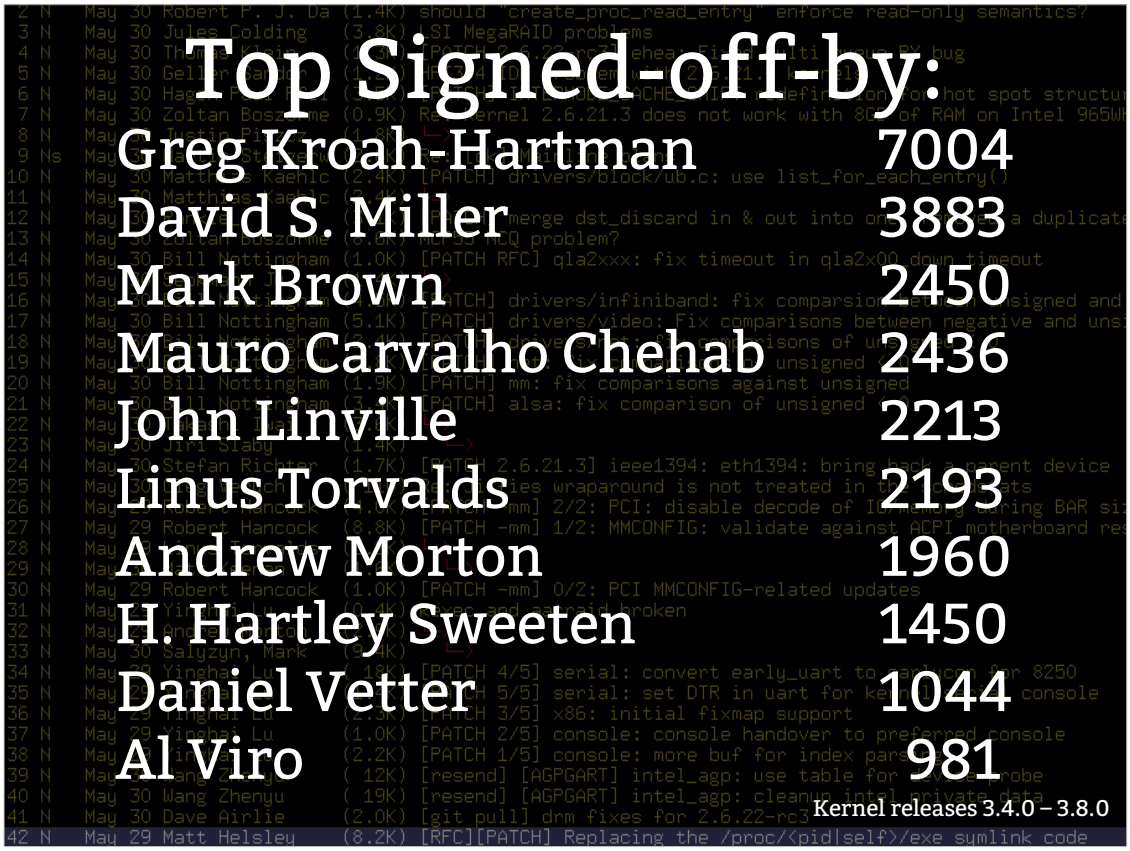
Top developers by quantity

H. Hartley Sweeten	1438
Mark Brown	642
Al Viro	553
Axel Lin	532
Greg Kroah-Hartman	505
Daniel Vetter	418
Johannes Berg	403
Bill Pemberton	394
David Miller	387
Sachin Kamet	387

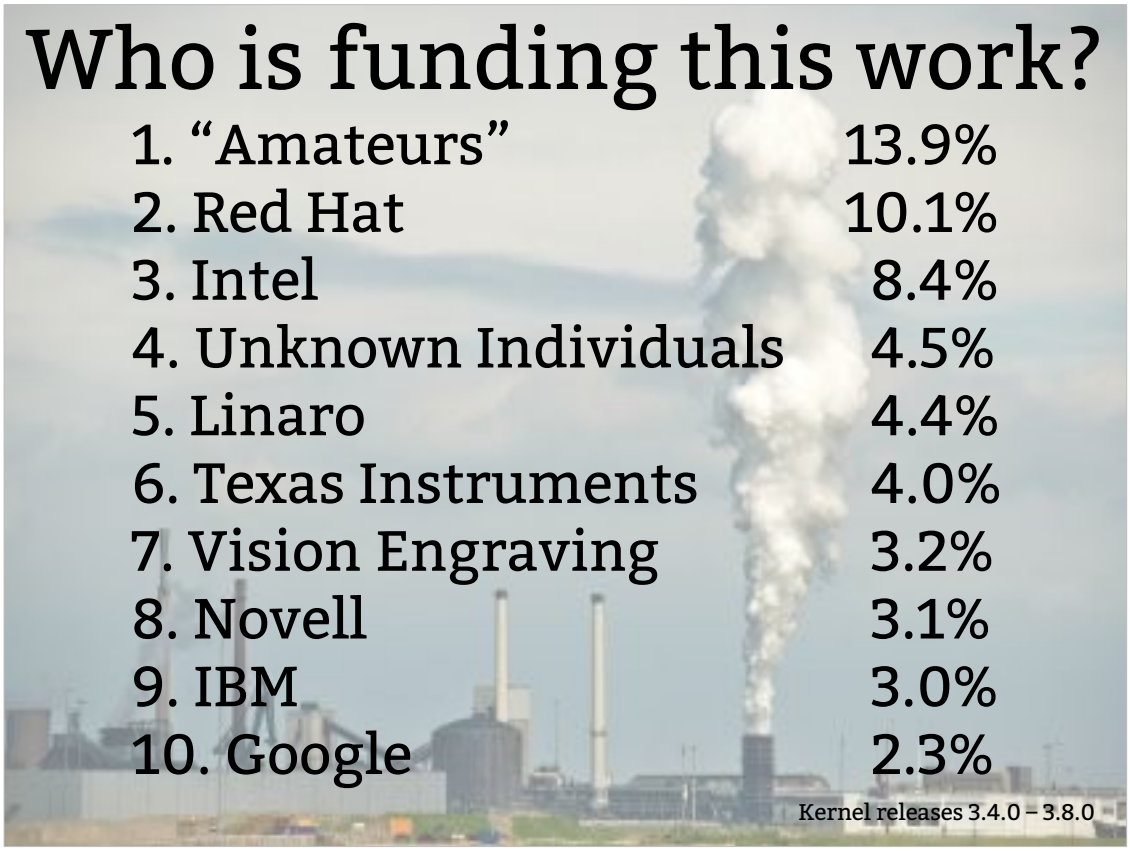
Kernel releases 3.4.0 – 3.8.0

Hartley - comedi
Mark – embedded sound
Axel – janitorial
Al – vfs and filesystem
Daniel – intel video
Johannes – intel wireless
Bill – janitorial
David – networking
Sachin – LINARO!

Greg – USB, staging, tty, etc.



Greg – driver core, usb, staging
David – networking
Mark – embedded sound
Mauro - v4l
John – wireless networking
Linus - everything
Andrew – everything
Hartley – comedi
Daniel – intel graphics drivers
Al – vfs



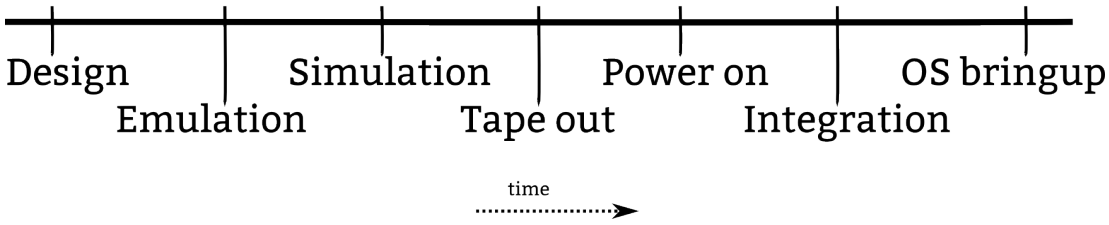
So you can view this as either 20% is done by non-affiliated people, or 80% is done by companies.

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

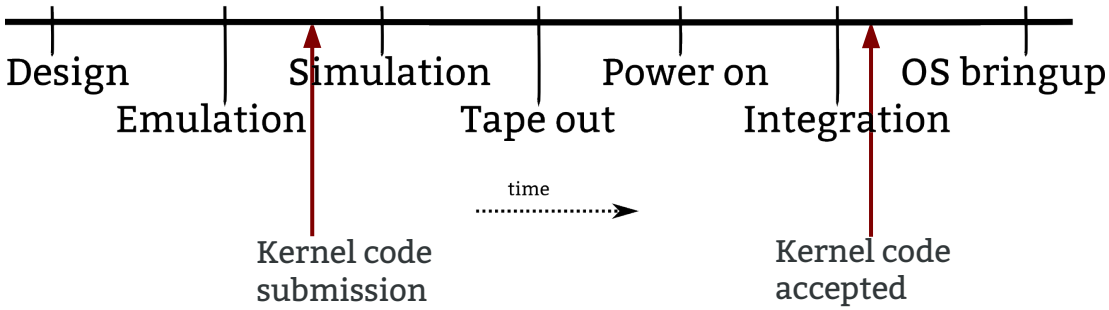
Why this all matters: If your company relies on Linux, and it depends on the future of Linux supporting your needs, then you either trust these other companies are developing Linux in ways that will benefit you, or you need to get involved to make sure Linux works properly for your workloads and needs.

Samsung 1021 patches
LF – 502 patches

Development Process



Development Process



“Working upstream saves time and money”

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

> Maintainers are like editors in the publishing industry.
>
> – David Miller

We work with developers to review, edit, suggest, reject, and hopefully, accept patches.

We also field bug reports, fix problems, and kick developers who are unresponsive.

When developers disappear, we end up taking over the maintenance of their code.

Every once in a while we get to do what we really love doing, new development.

Patches I received in a 2 week period

So, let's look at one of these subsystem maintainers. I maintain the USB, driver core, tty, staging, and a few other various parts of the Linux kernel.

This 2 week timeframe is when we had our big merge window, when all of the subsystem maintainers sent patches off to Linus. During this time frame, no core kernel developer sends new stuff to subsystem maintainers, as they know they are busy, and nothing that gets sent can really be looked at until after the merge window closes.

So, almost all of the patches I got in the past 2 weeks were not from developers that do a whole lot of kernel work, nor were the, for the most part, large patches with new things being proposed for the kernel.

Patches I received in a 2 week period

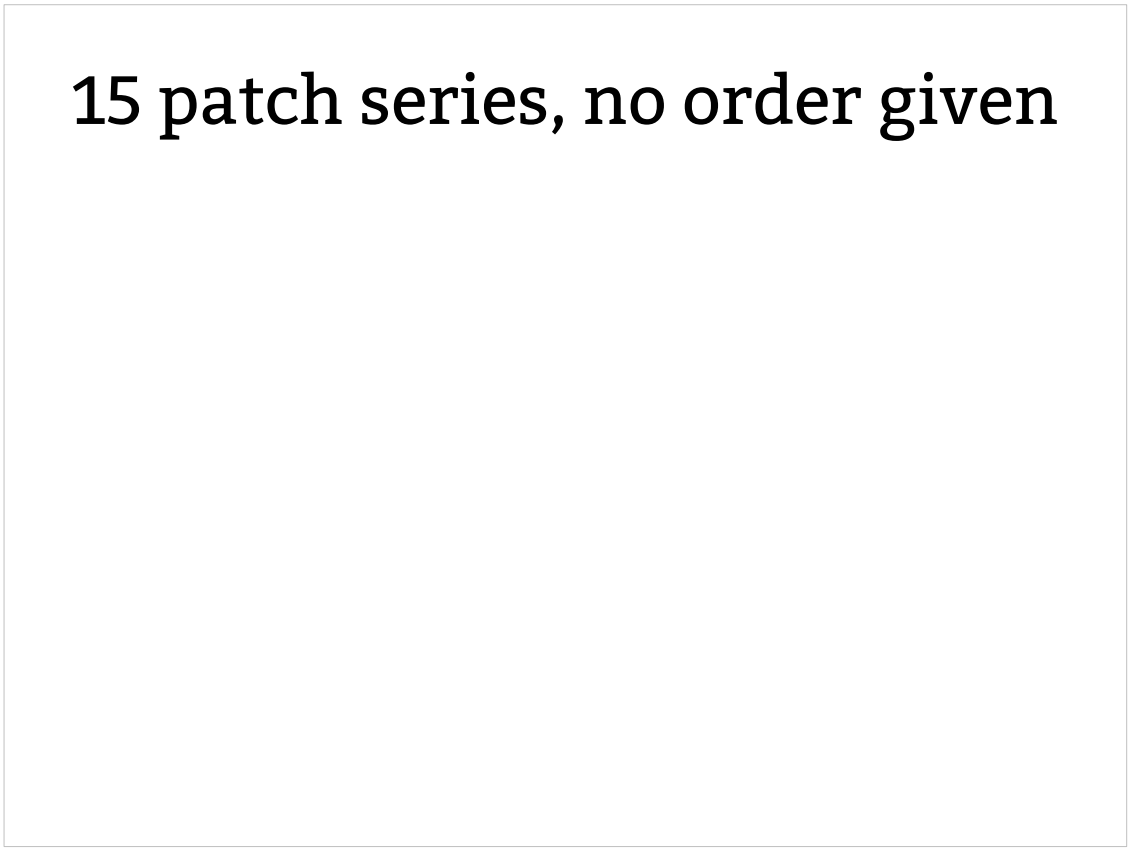
487

Yeah, that's the number of patches I got during the "slow" period of the kernel development cycle. This does not include the number of messages around those patches as other developers commented on them, or other various things about those patches (like "have you applied my patch yet?" messages.)

Now the large majority of these patches at first glance look just fine. But I took a closer look at them, and here's a short list of the problems in the patches that were sent to me.

Subject: [PATCH 48/48] ...

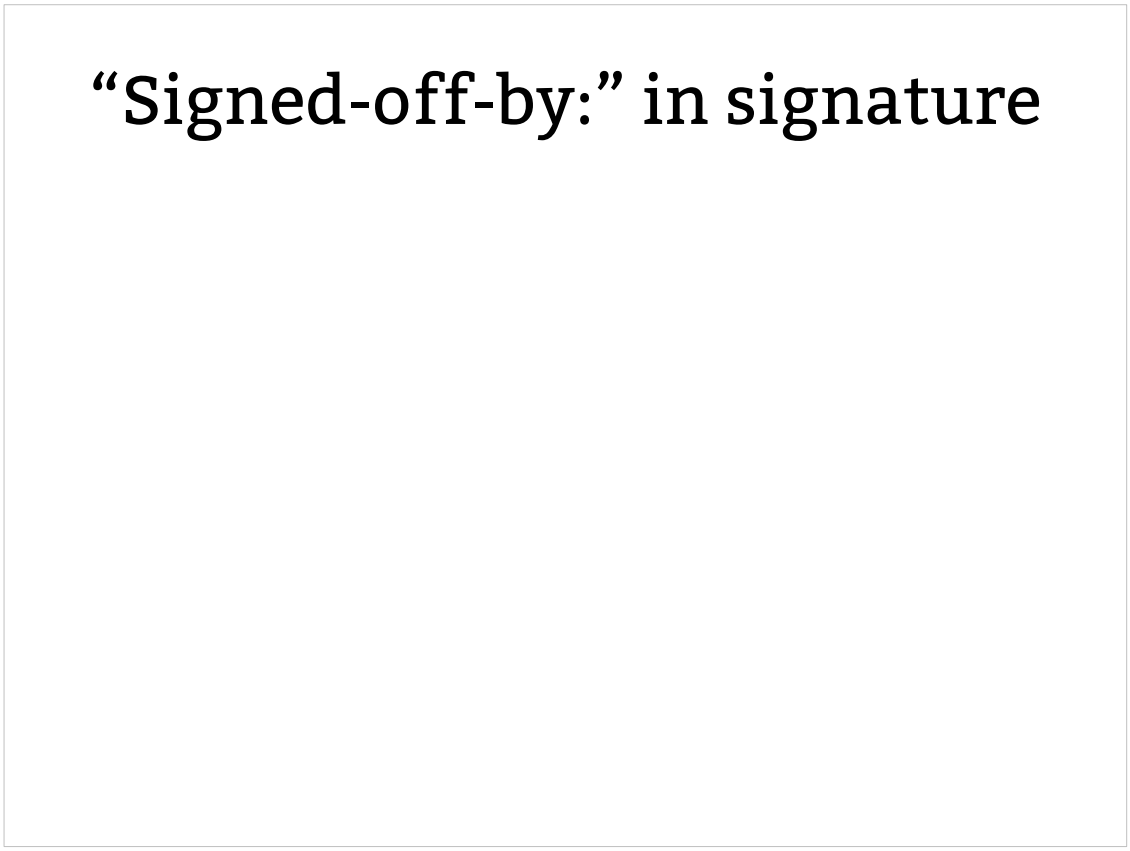
There were no 47 previous patches sent.



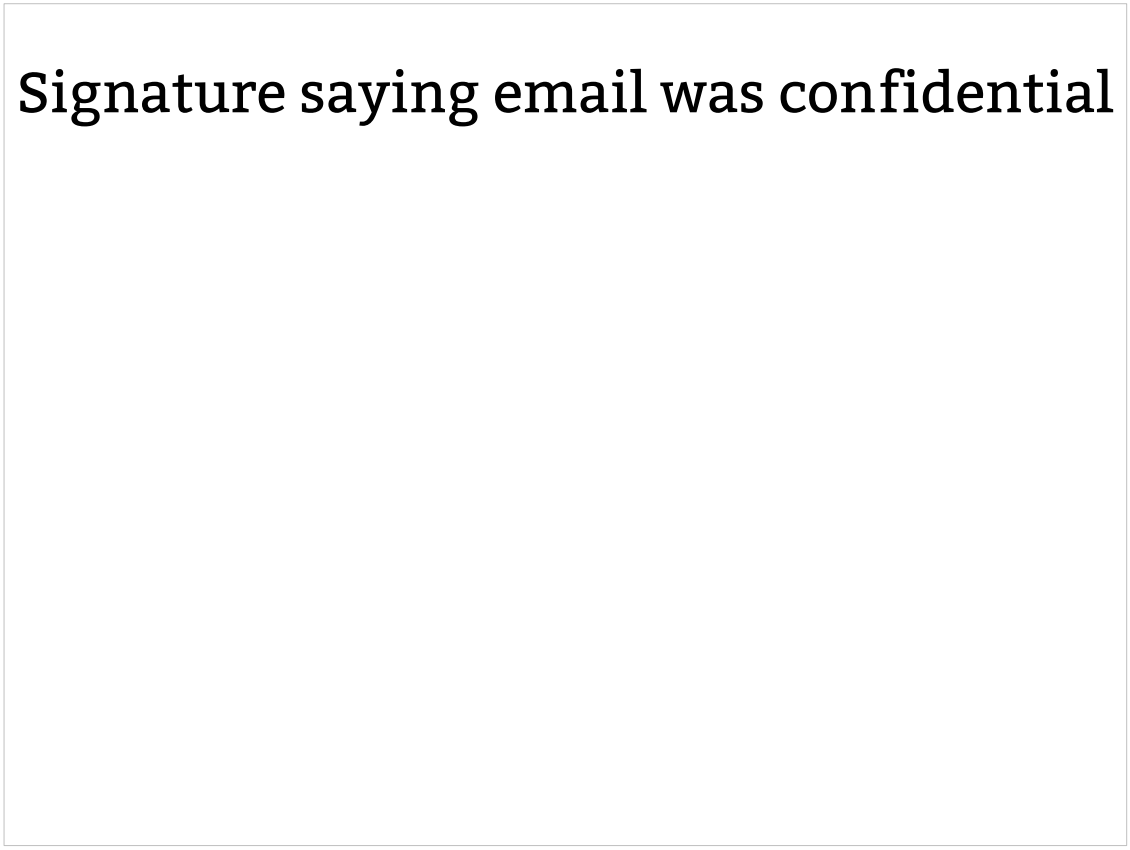
Am I supposed to guess?

Number 2 never showed up.

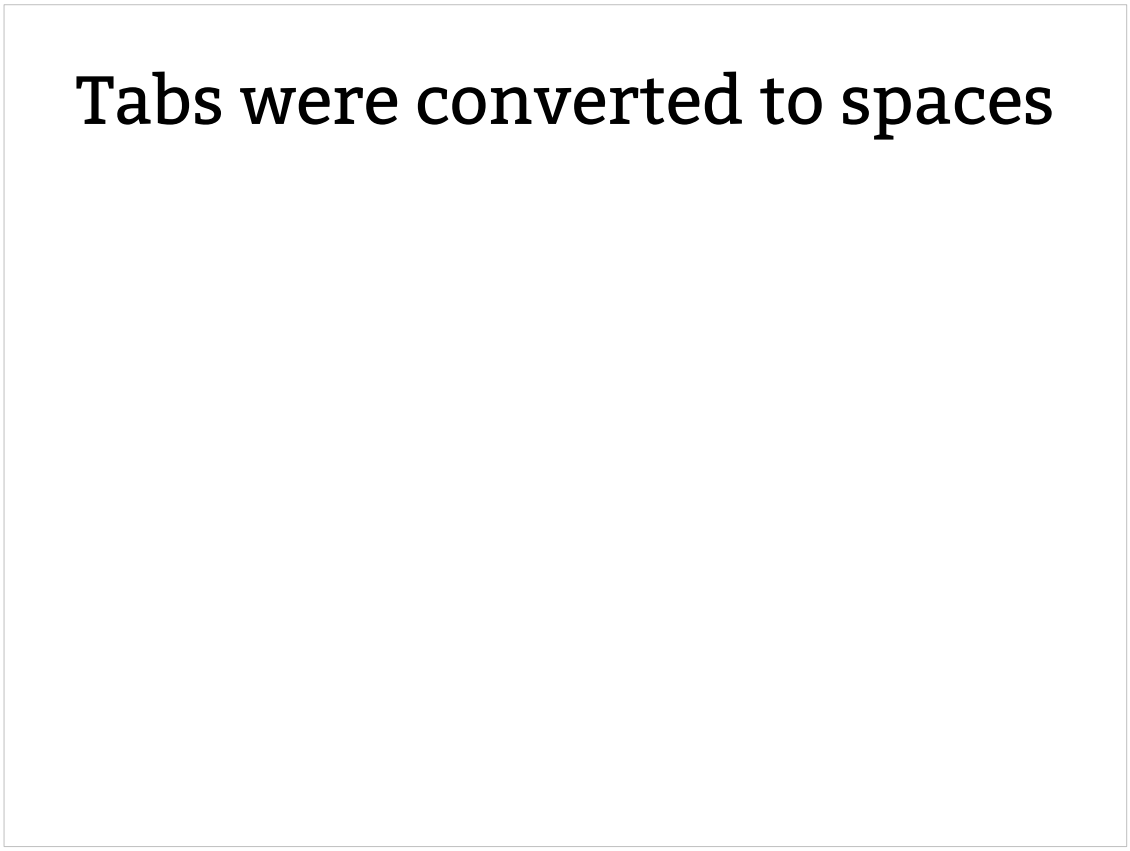
“Signed-off-by:” in signature



This would require me to hand edit the patch before I could apply it.



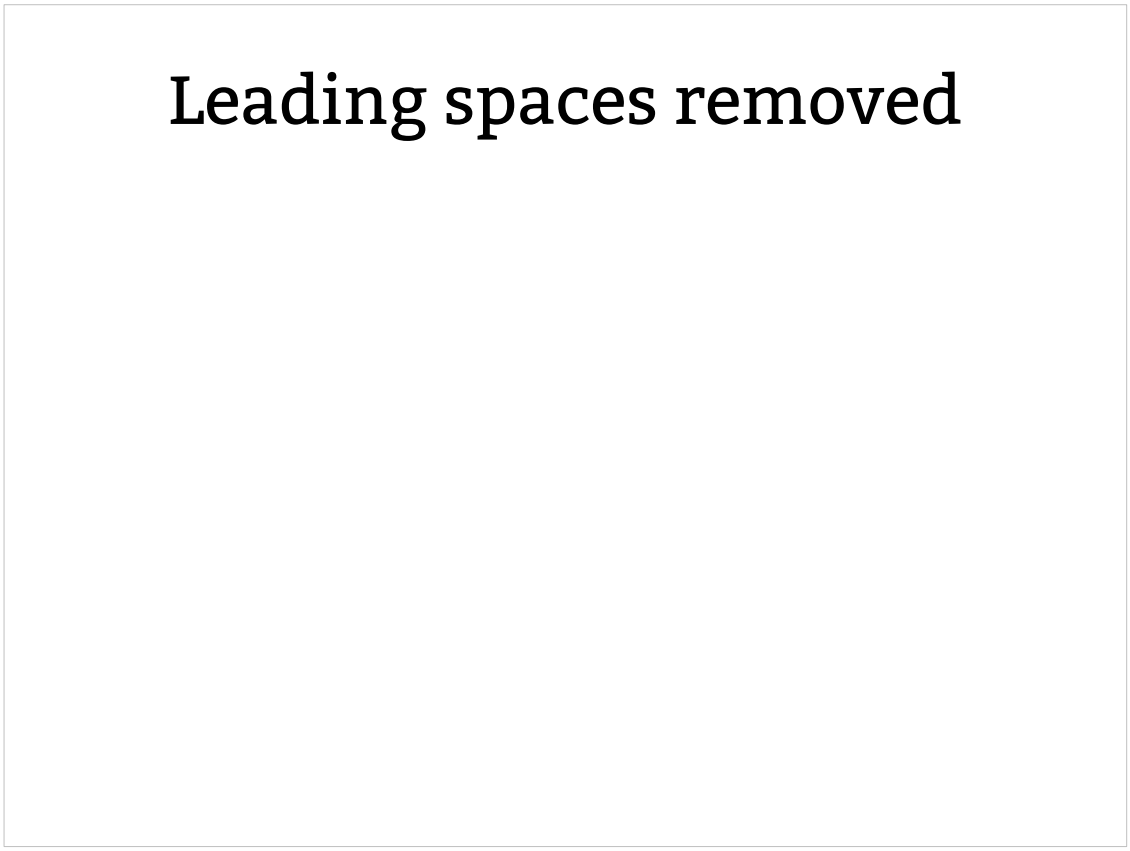
That kind of goes against how you are supposed to be sending Linux kernel patches out to the world.



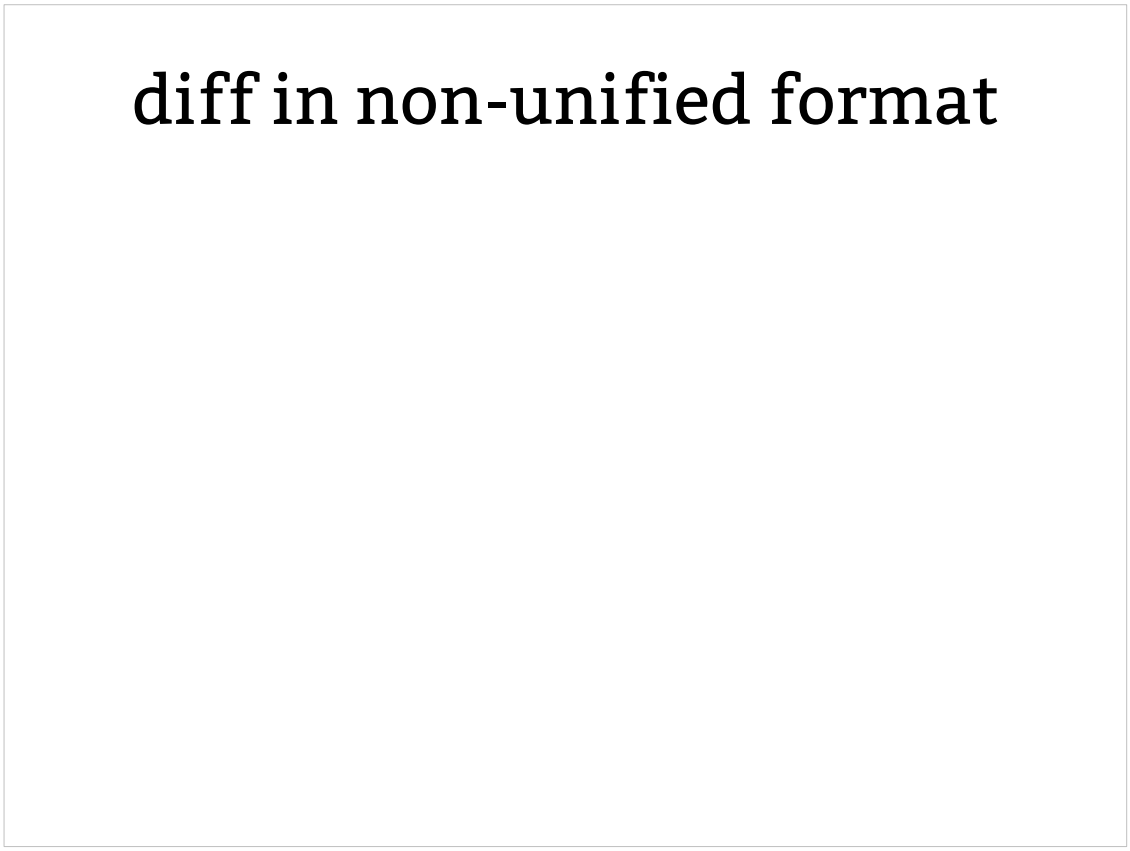
This makes applying the patch impossible.

Exchange does this for you, if you are working for a corporation that has an Exchange server, do what IBM, Intel, and Microsoft have done in order to be able to contribute to Linux kernel development, have a Linux box somewhere in the corner that your developers use as a mail server to send patches out from.

Huawei is the only company that I know of that successfully sends kernel patches through an Exchange server, which is amazing, I really don't know how they do it.



This also makes applying the patch impossible. I end up editing a lot of patches by hand, cursing all the while, just to get them to apply because of broken email servers and clients.

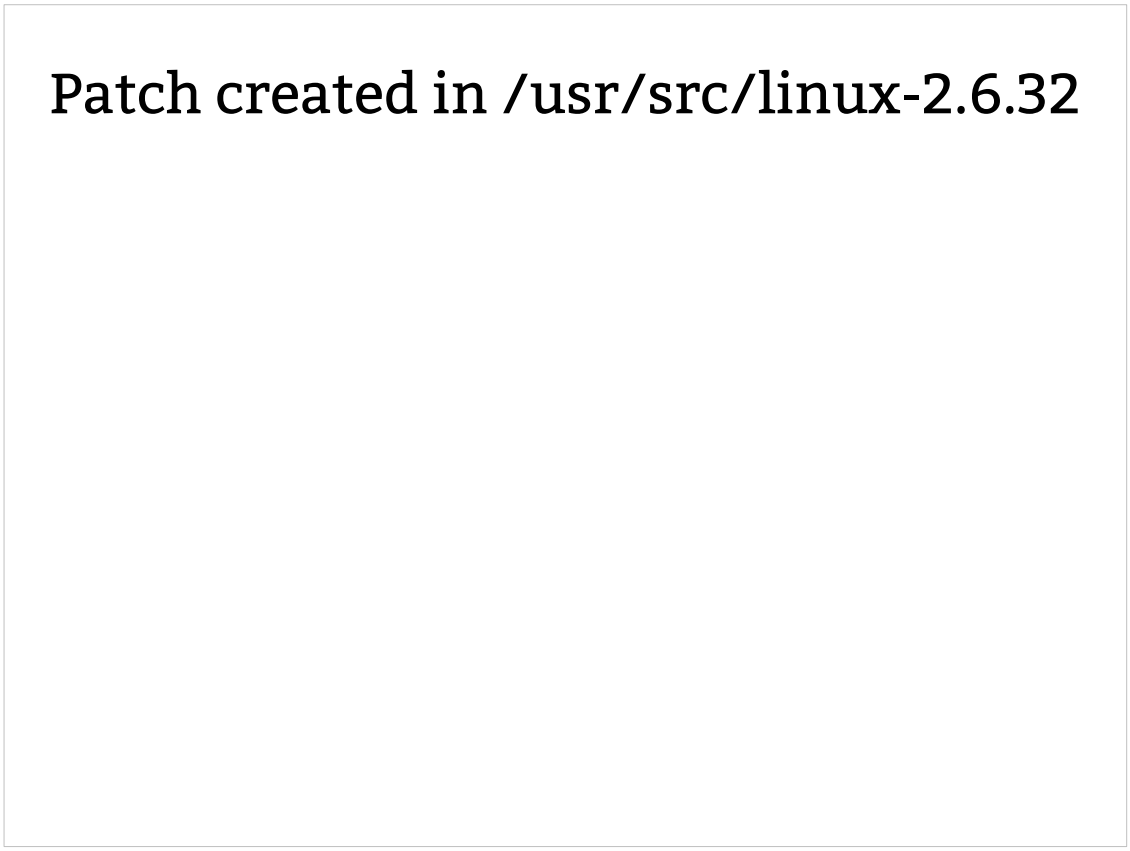


I honestly didn't know that diff could still create output in this format anymore, I assumed that as no one ever found it useful, it wasn't used anymore.

Patch created in driver directory

Patches need to be created in the root of the kernel source tree, as that's where I have to be in order to apply them properly.

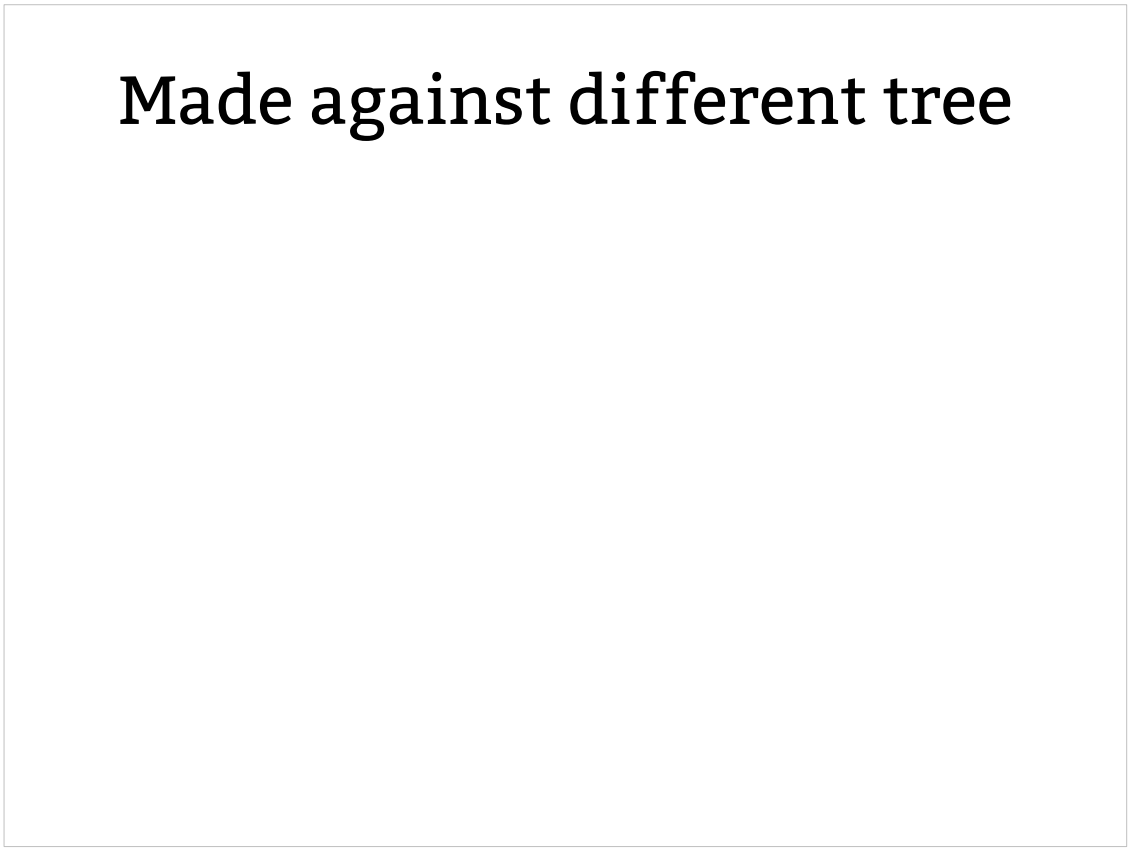
This seems to happen a lot to first-time patch submitters, it's a very common problem.



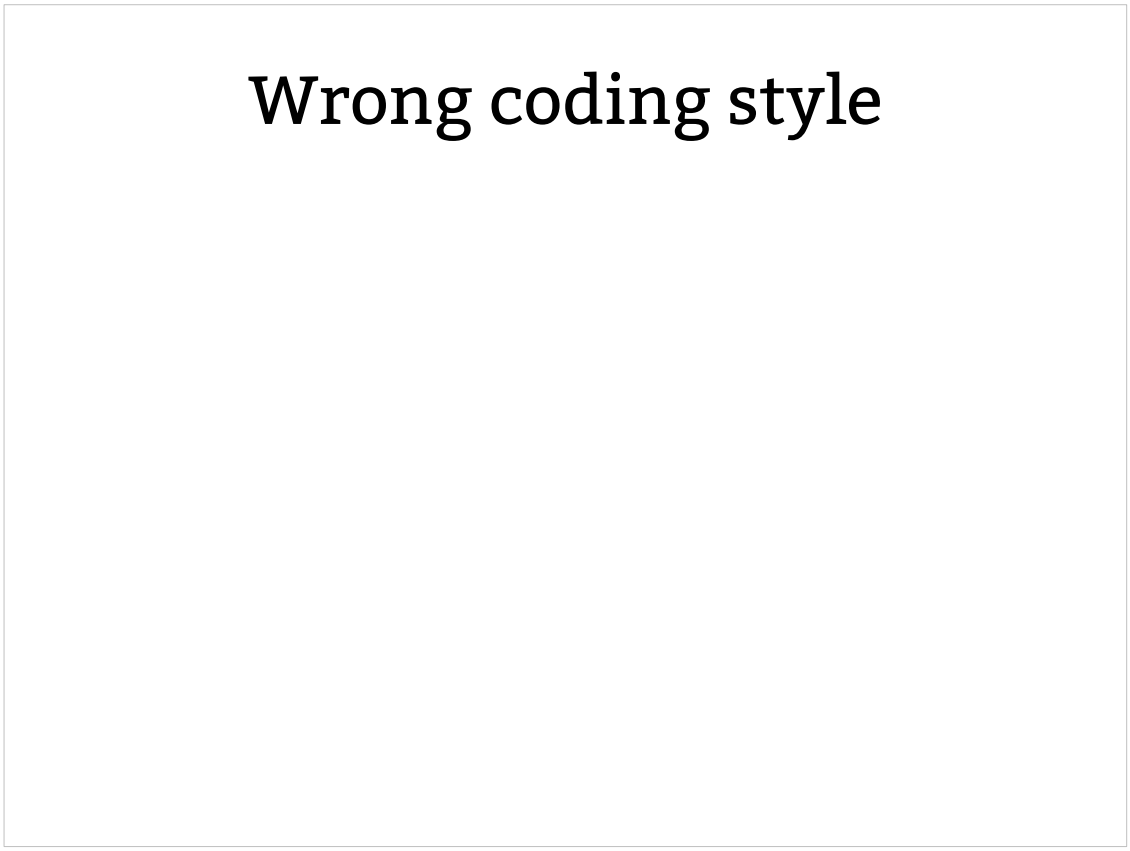
How many different problems can you see here in just this one example?

Old and obsolete kernel version, full path to root, developer doing kernel work as root, probably more.

Made against different tree



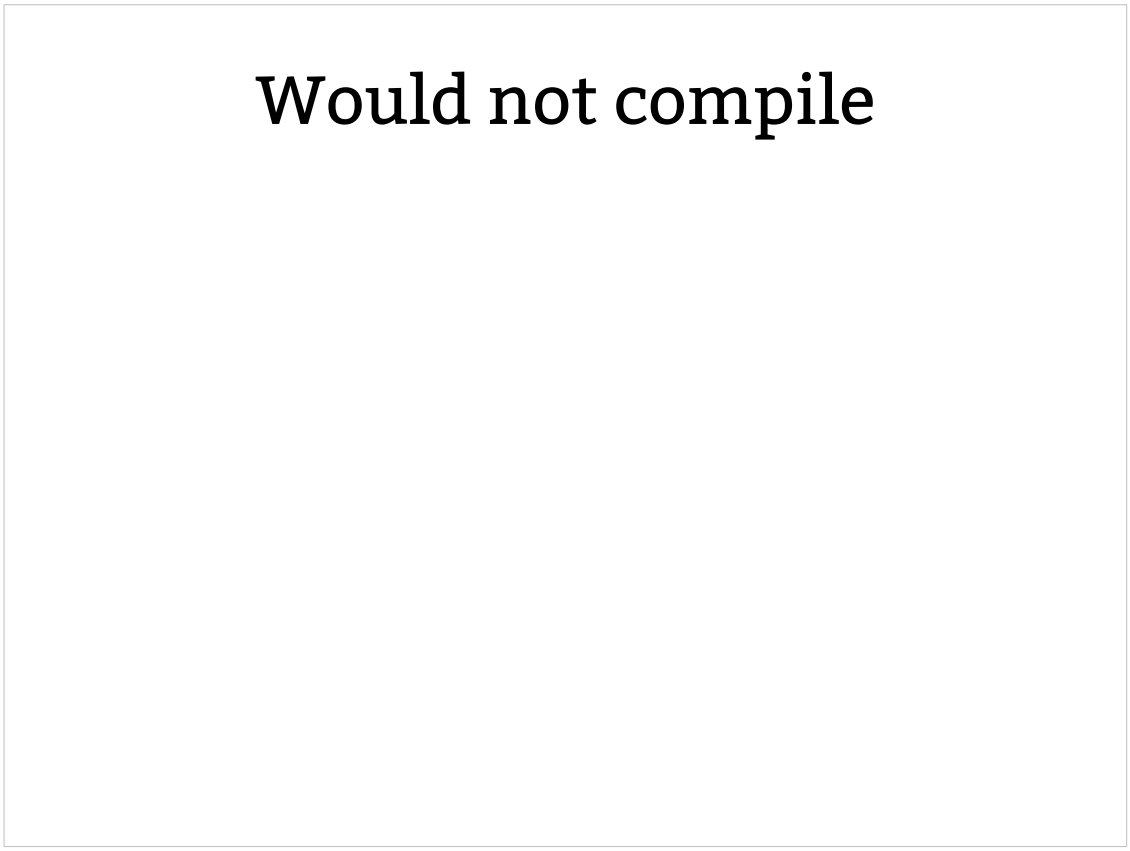
Someone made a patch against the scsi subsystem development tree when sending me a USB patch. Why they thought that was a good idea I have no idea.



There's no excuse for doing something like this anymore, we have automated tools that fix this up for you.

Wrong coding style, and acknowledged it

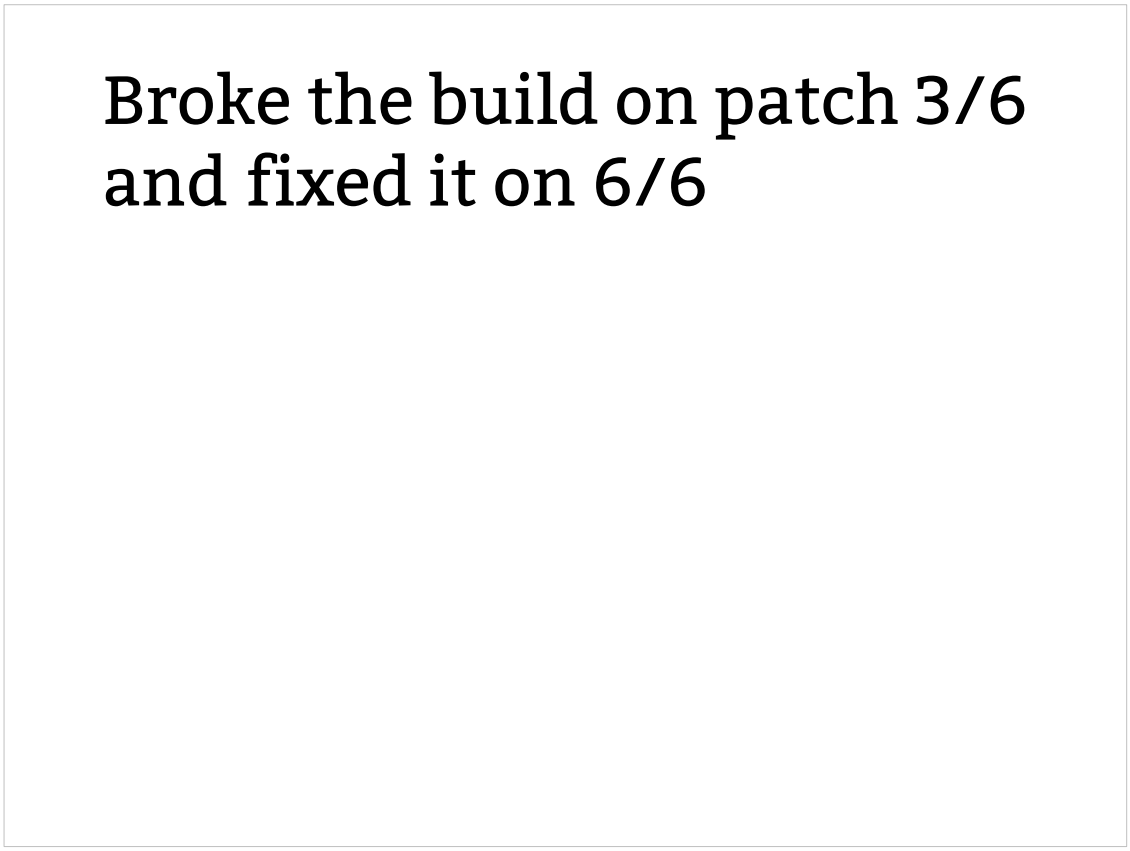
At least in this patch, the author knew they were doing something wrong, It's just that they thought they were more important than the 3000 other kernel developers and didn't have to play by the rules of everyone else.



Just looking at the patch it was obvious that it had never been compiled, and sure enough, the compiler spit out a bunch of errors.

Broke the build on patch 3/6

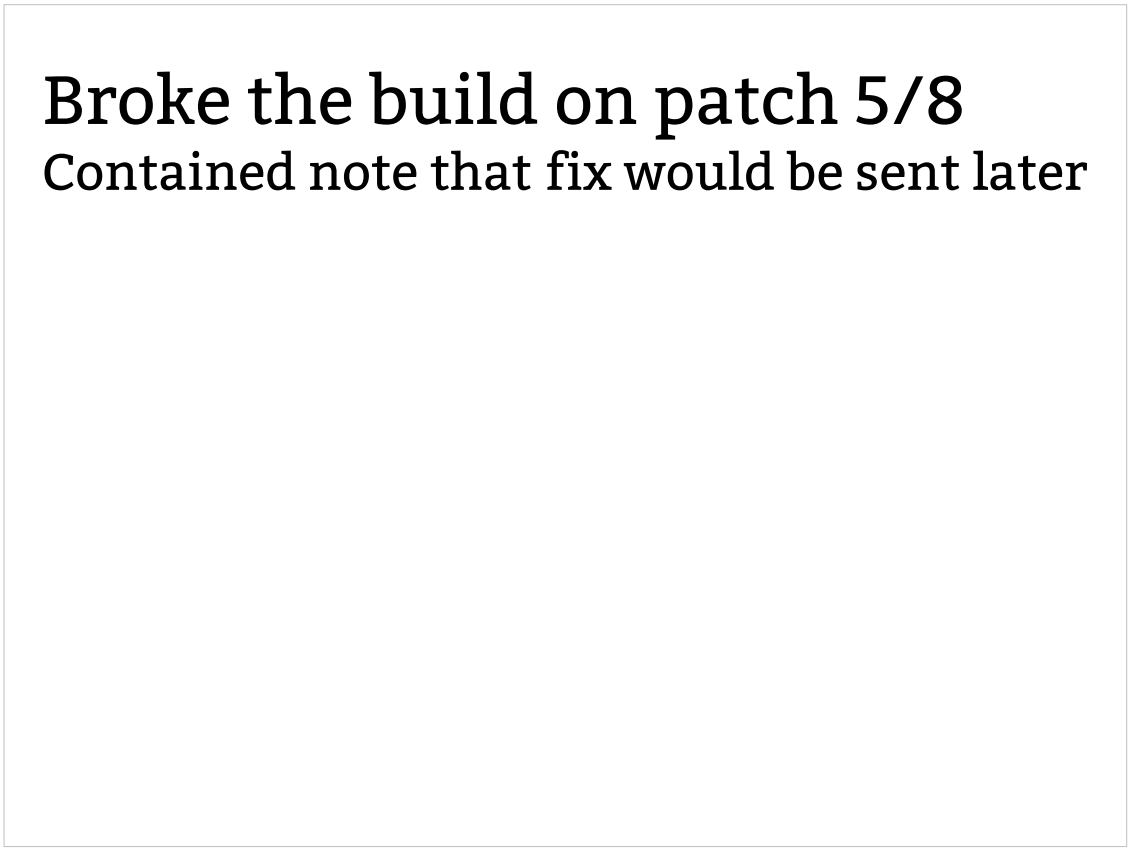
This was a series of patches, and the build broke on the 3rd patch that was applied.



But, I looked closer, and the developer at least realized this, and fixed it in their last patch in the series, thinking that all was now good, as it didn't really matter that for the past 3 patches, the build was broken.

Broke the build on patch 5/8

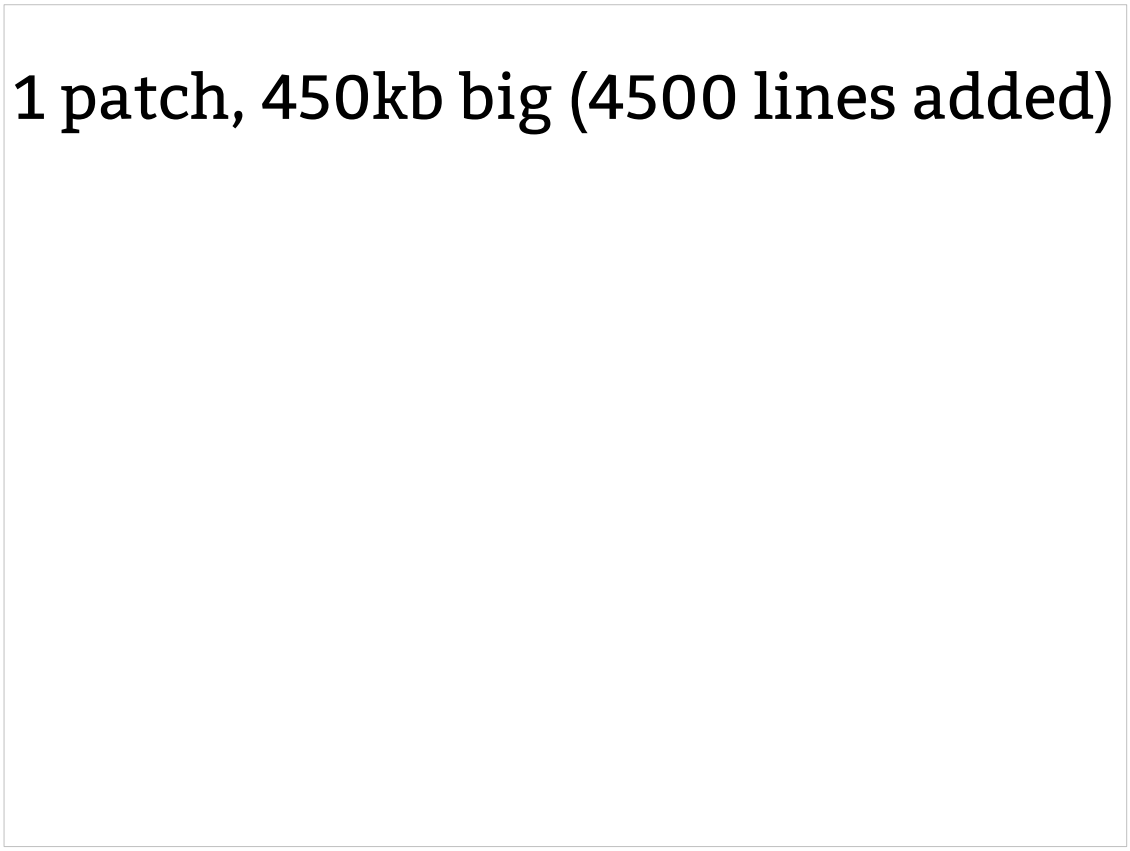
There was another patch series that also broke the build in the middle of it.



But this one was better, it had a note saying that they knew the build was broken, and they would fix it up later, at some unknown time in the future, but these 8 patches should be accepted now anyway.

Patches that had nothing to do with me

Now I know I maintain a lot of different parts of the kernel, but for some reason someone sent me a patch for the x86 core code, copied to no one else, thinking that I was the one that could accept it.

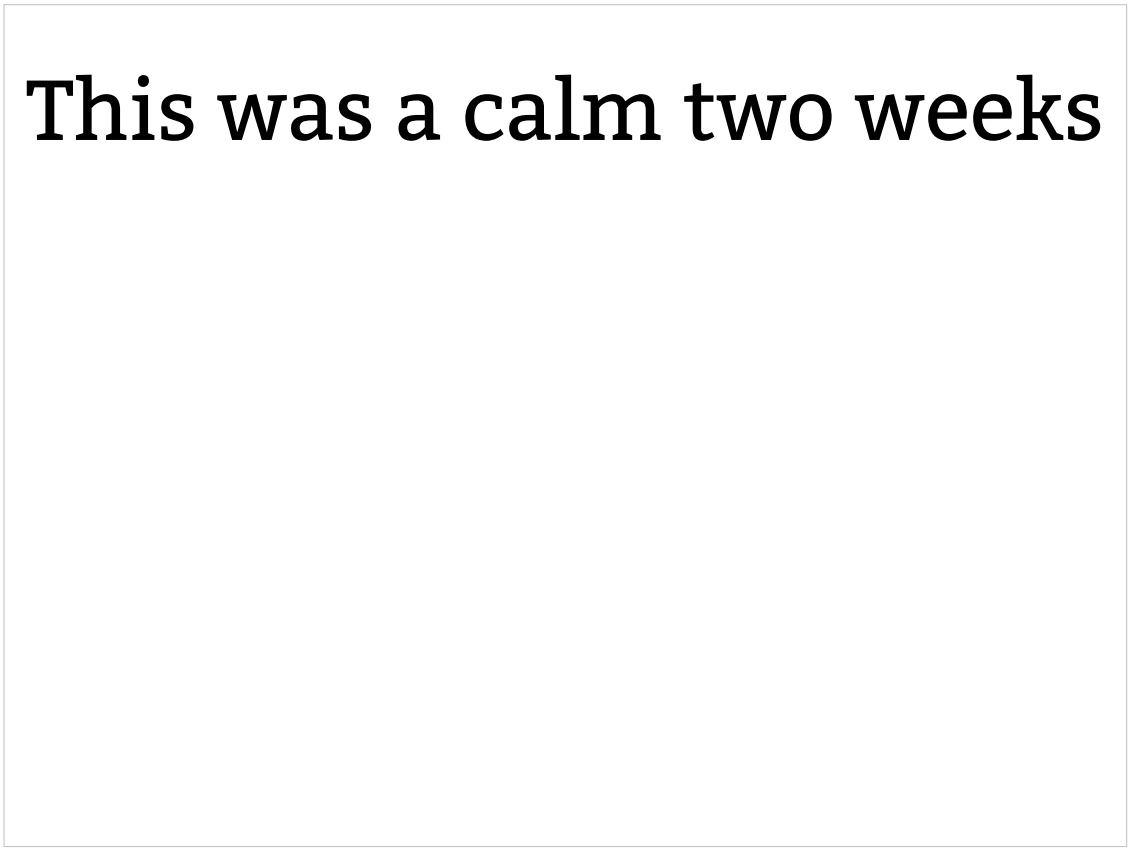


Luckily, another developer told the author that this was too big and needed to be broken up into smaller pieces before anyone would review it. And then, three different developers went and reviewed it anyway, so I don't think the author learned that lesson at all.

Obviously wrong kerneldoc

kerneldoc is the format that you can write comments in the code and get them included in the kernel api documentation that is automatically generated. When you get the format of it wrong, the tools will tell you.

Here was someone who was trying to write documentation, but got the format wrong, and hadn't even run the tools to see if it was generated properly.



Now, I'm not asking you to take pity on me, just realize that this is the level of incompetence that every single one of those 700 developers encounter all the time.

So when you think we are acting grumpy, remember, how would you act if you had to deal with this all of the time?

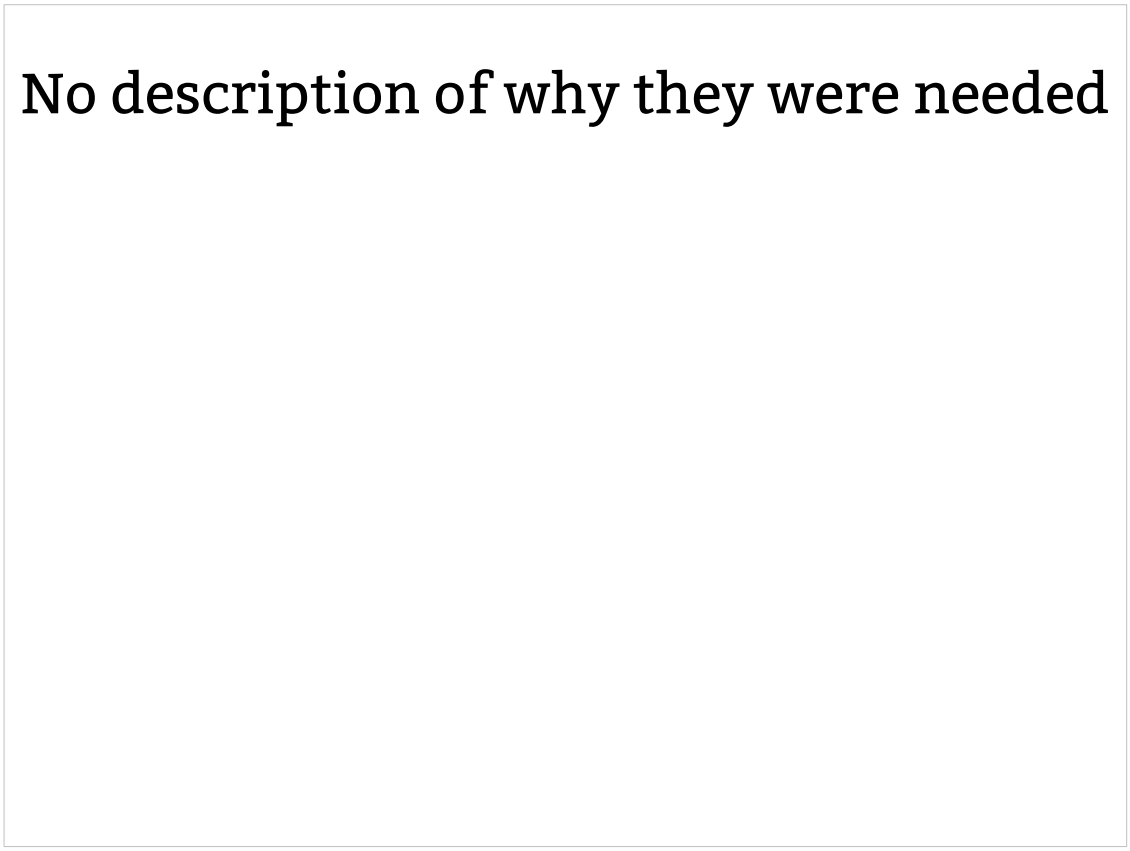
Let's get back to what the goal is here. You want to create a patch that is accepted as it does something that you want to do in Linux. The maintainer wants to reject it.

Case study of a Linaro patch submission

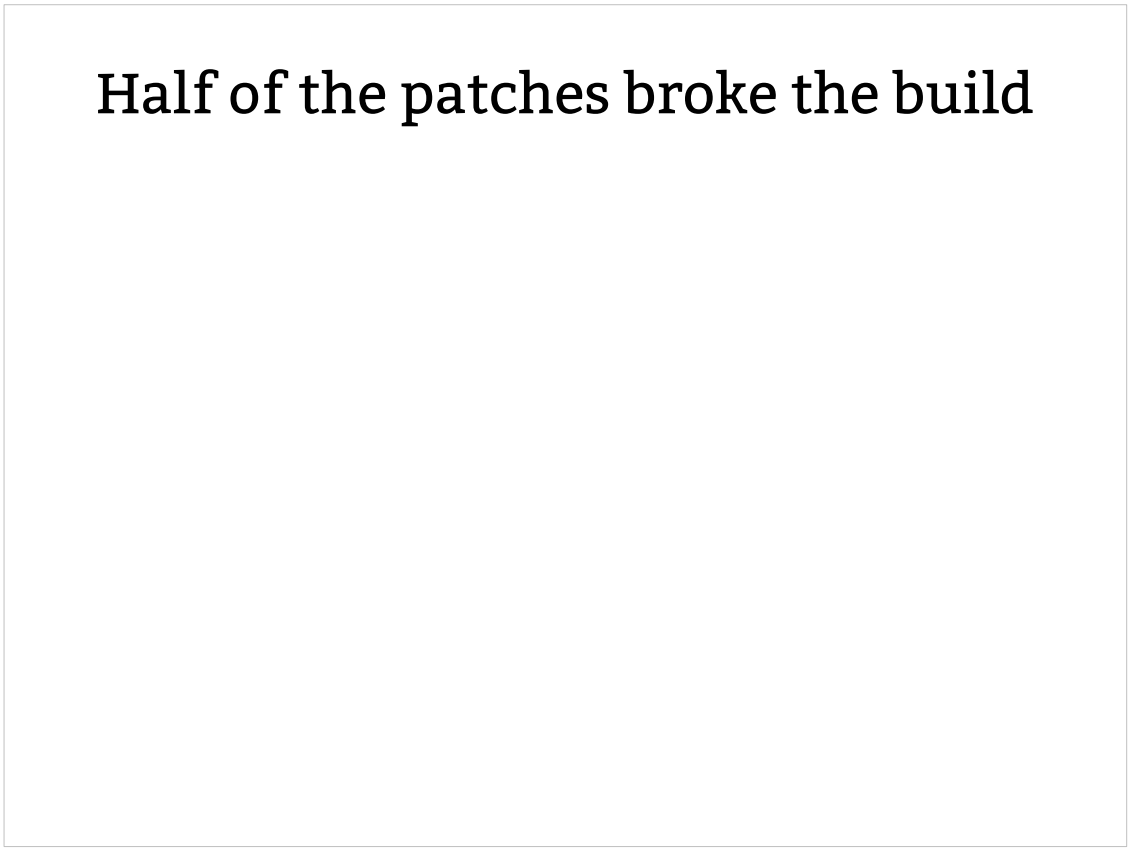
8 patch series for USB

Let's look at a patchset from Linaro that was sent to me over the past month.

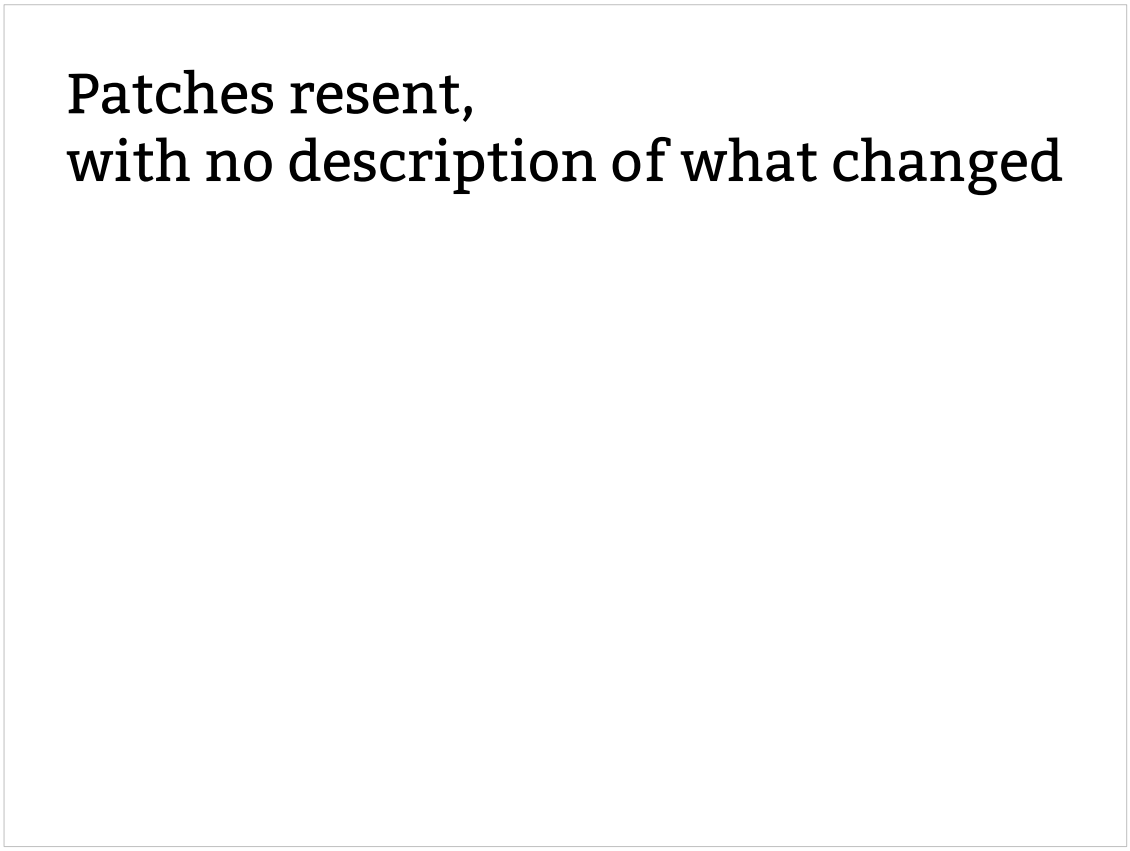
You all should really know better than to do things this badly.



1 sentence description of what the patches did, but anyone can read that.



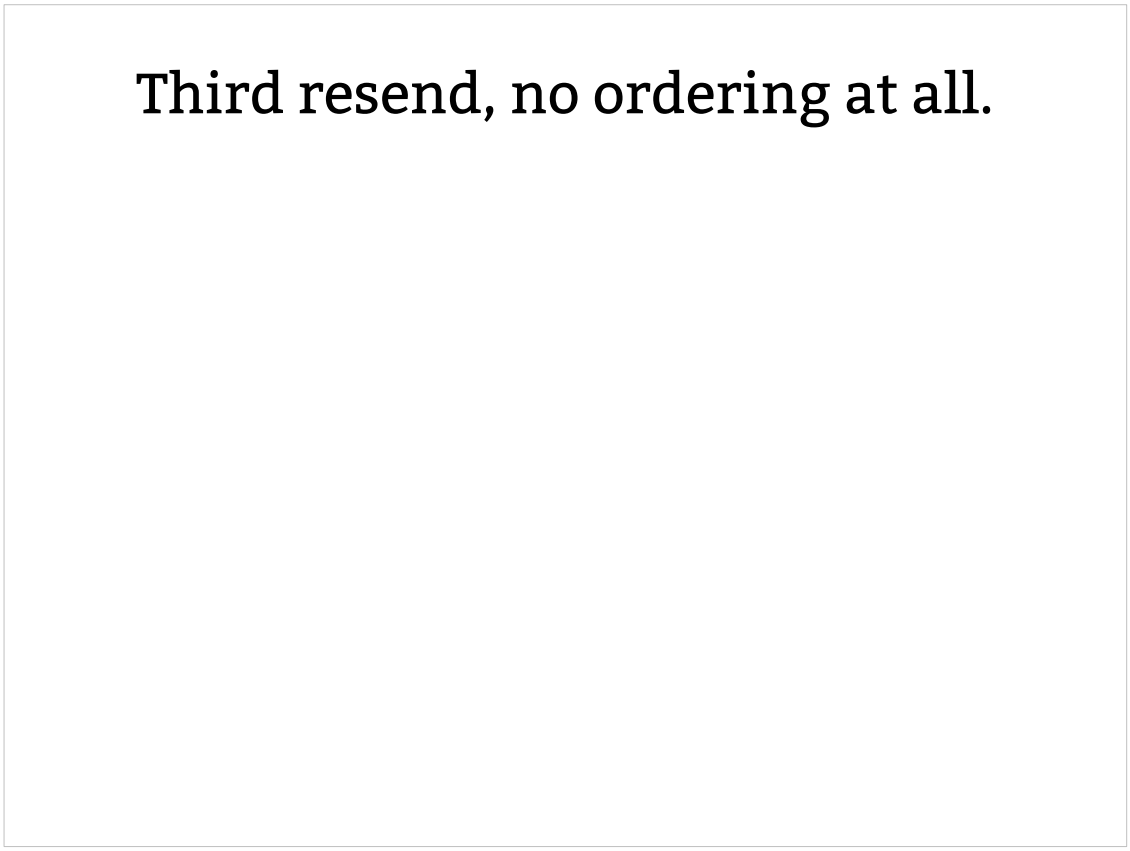
Obviously not even tested.



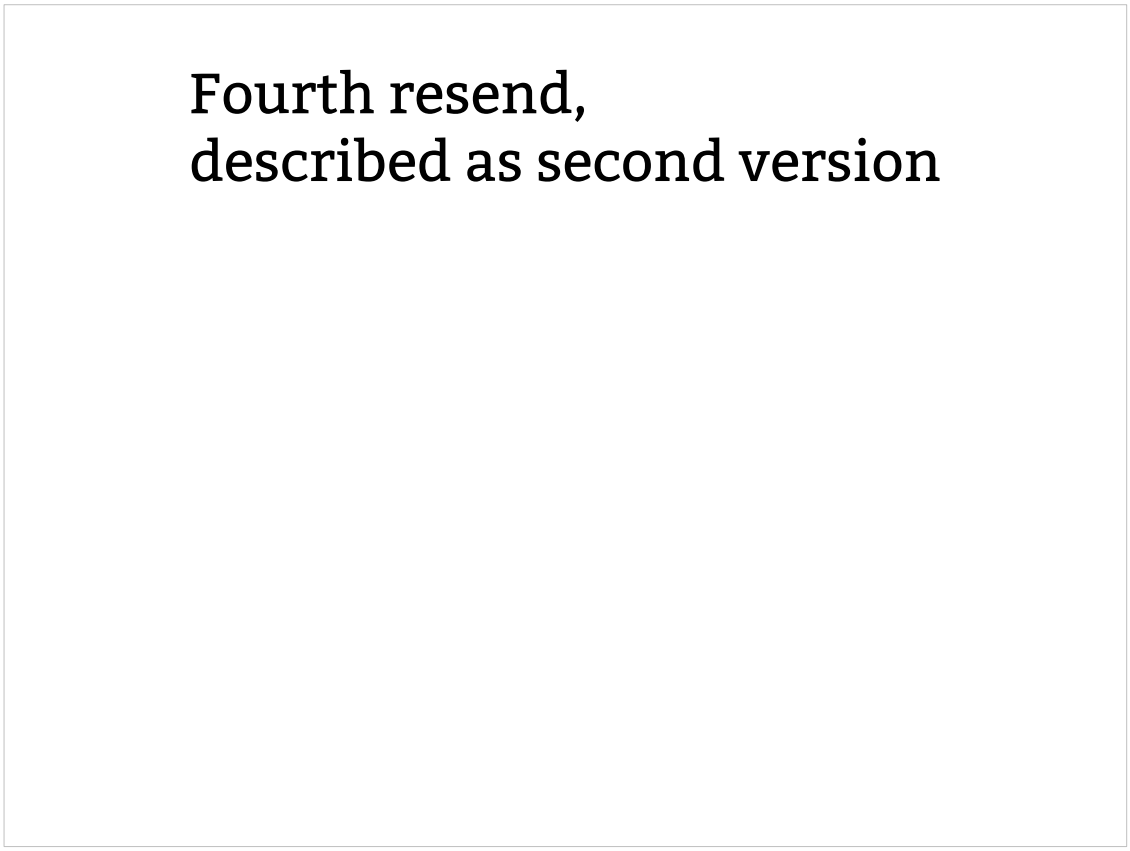
I have no idea if they were fixed or not.

Replacments sent again, individually, out of order

Someone else from Linaro stepped in, told them to fix this all up and resend properly.



Rejected.



Obviously not true, am I stupid?

Insisted that 2 of these HAD to go into 3.9-rc1.

We are now days away from the merge window closing for patches.

We are now days away from the merge window closing for patches.

The Linaro senior developer stepped in, and said that these two were ok to apply.

Those two required a follow-on patch to fix the Kconfig descriptions

Ok, people forget to update the documentation, that's normal.

Patches broke all non-DT systems

1 week after my merge window closed, I reverted these.

Driver maintainer returned
from vacation and reviewed them

Driver maintainer returned
from vacation and reviewed them

Every single one was broken

Driver maintainer returned
from vacation and reviewed them

Every single one was broken

“...you can answer at lot of questions like this for yourself very easily, simply by reading the source code.”

I will not accept a patch directly from this developer, for a very long time

Do you blame me?

```
static void tty_slave_release(struct device *dev)
{
  struct tty_slave *tts = to_tty_slave(dev);

  kfree(tts);
  /* Test code to see if slave device get released */
  BUG();
}
```

It's not just Linaro.

This was submitted by a company that has much more experience than Linaro in kernel development.

It was asked to be merged.

It was obviously never actually tested.

This was the 5th version of this patch.

It is in my self-interest to ignore your patch

Seriously. It's easier for the maintainer to not accept your code at all. To accept it, it takes time to review it, apply it, send it on up the development chain, handle any problems that might happen with the patch, accept responsibility for the patch, possibly fix any problems that happen later on when you disappear, and maintain it for the next 20 years.

That's a lot of work that you are asking someone else to do on your behalf. You are asking someone who doesn't usually work for your company, who probably lives in a different country, who you have never met in person, to assume responsibility for your work, and to do extra work on top of the normal work they do in the kernel every day.

So you can see how it's in my interest to ignore your patch. And it's in your interest to keep me from ignoring it, because you want it accepted.

Give me no excuse to reject your patch

So your goal is, when sending a patch, is to give me NO excuse to not accept it. To make it such that if I ignore it, or reject it, I am the one that is the problem here, not you.

What can you do to keep me from rejecting your patch outright

.

First off, don't do any of the things I listed above, that's obvious, right? But that's a "do not do" list, how about a list of what to do:

What I will do for you:

So, finally, you created the perfect patch series, took all review into account, and sent it correctly, without corrupting the patch.

What should you expect from me, the subsystem maintainer?

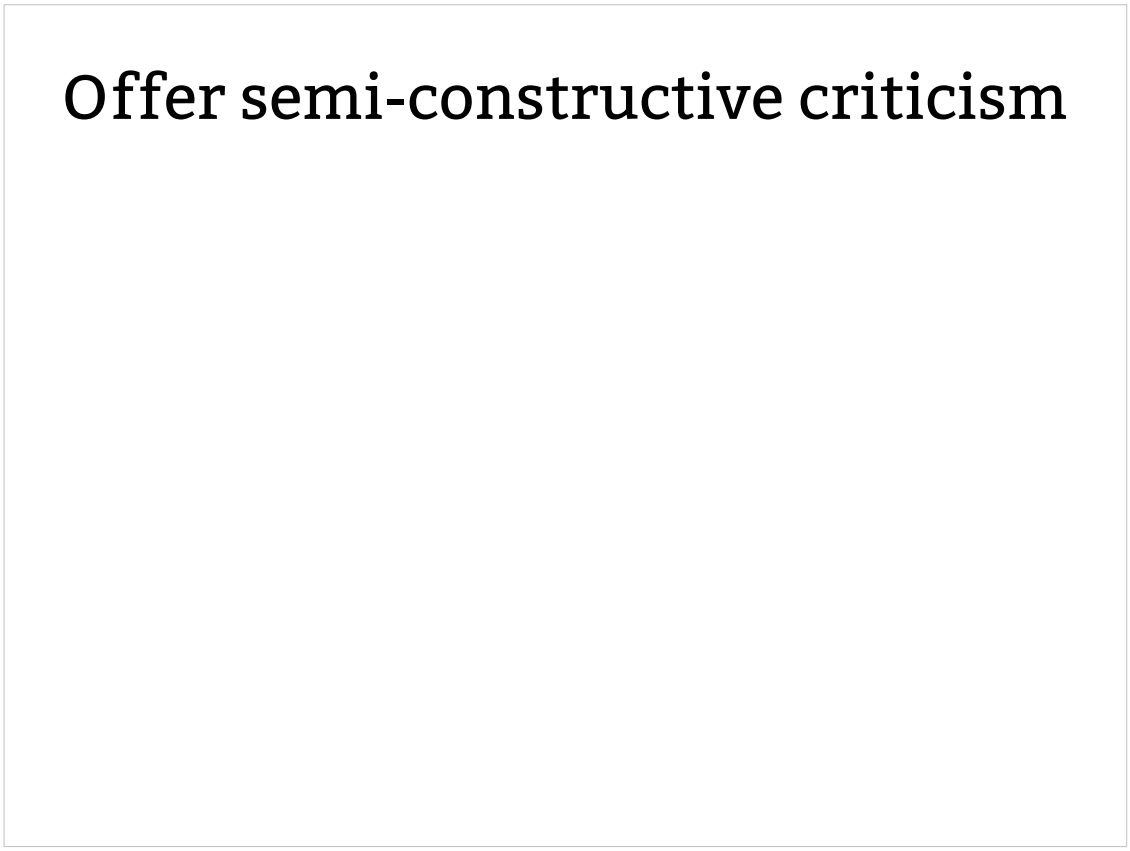
Review your patch within 1-2 weeks

Some subsystem maintainers try to get to patches even faster than this, but with travel and different conferences, the best that I can normally do is about 1-2 weeks.

If I don't respond in that time frame, just ask what is going on. I have no problem with people asking about their patch status. Sometimes patches end up getting dropped on the floor accidentally, and if I'm being slow I have no problem with being called on it, so don't feel bad about checking up on it.

But please wait 1-2 weeks, don't be rude and send a patch at night, and then in the morning send a complaining email asking why it wasn't reviewed already. This happens more than you want to know.

Offer semi-constructive criticism



I can't always promise constructive criticism, but I'll try my best.

Let you know the status of your patch

I have a set of scripts that I got from Andrew Morton that will email you when I apply your patch to one of my development trees saying where it has been applied and when you can expect to see it show up in Linus's tree. There is no reason that all kernel maintainers shouldn't do this, and it's nice to see that more and more are.

But, I know from personal experience, there are maintainers in this room that I send patches to and I never know what happens to them. A few months later I will see them show up in Linus's tree, usually after I forgot about them.

That's not acceptable, and you should not allow this, push back on your subsystem maintainer to use something like this, to keep you informed. Andrew's scripts are public, as are my variations of them, for everyone to use.

> “Publicly making fun of people is half the fun of open source programming.
>
> In fact the main reason to eschew programming in closed environments is that you can't embarrass people in public.”
>
> – Linus Torvalds

Linus said this after I wrote a small rant about some truly horrible Linux kernel driver code that I found online.

It had all sorts of "this code is never to be included in the Linux kernel" warnings all over it, despite it being a Linux kernel driver. And in reading the code, it was obvious why the author never wanted it in the kernel, it was one of the worse things I had ever seen, and that says a lot. After I complained about it, I felt bad, as someone had obviously spent a lot of time on it, but Linus replied with the above quote.

And as always, it turns out that Linus is right.

> “Publicly making fun of people is half the fun of open source programming.
>
> In fact the main reason to eschew programming in closed environments is that you can't embarrass people in public.”
>
> – Linus Torvalds

If that author had ever thought that the code would have been posted publicly, they wouldn't have written such crap. That's what maintainers and public code review is really for in the end, keeping the crap out of the Linux kernel, which benefits everyone involved.

So while it seems that we kernel developers can be a real harsh bunch of people, it is because of that harshness that Linux is as good as it is.

You want us to be tough, because it makes you better programmers in the end, knowing you will have to defend your code.

And that is why I love doing this work, it makes everyone involved produce the best possible code, which in the end, is what matters the most.

Obligatory Penguin Picture