Linux Kernel Maintainers

Greg Kroah-Hartman
gregkh@linuxfoudation.org

Why are they so grumpy?

What you can do to avoid this.

What maintainers owe you.

2,833 developers
373 companies

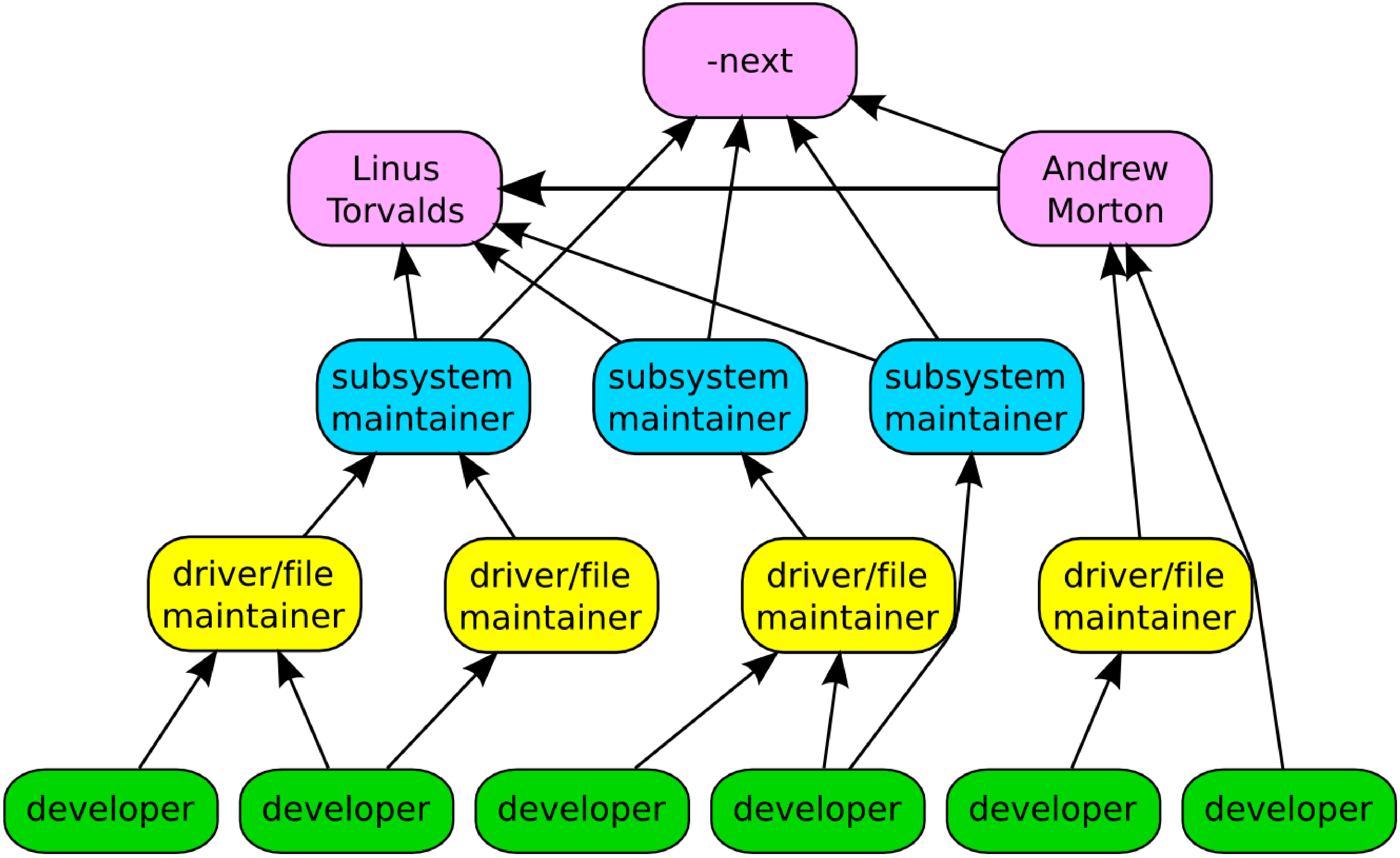
Kernel releases 3.0.0 – 3.4.0
May 2011 – May 2012

5.79 changes per hour

Kernel releases 3.0.0 – 3.4.0
May 2011 – May 2012

7.21 changes per hour

3.4.0 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

Patches I received in the past 2 weeks

Patches I received in the past 2 weeks

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

It is in my self-interest to ignore your patch

Give me no excuse to reject your patch

Proper coding style

scripts/checkpatch.pl clean

Sent to proper people and lists

Sent to proper people and lists

scripts/get_maintainer.pl

Proper Subject:

Proper changelog comment

Description of WHY it is needed

Small incremental change

“obviously” correct

Which tree it was made against

If multiple patches, state the order

Has to build properly

Make sure it works, if possible

Don't ignore review comments

Don't resend without saying why

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

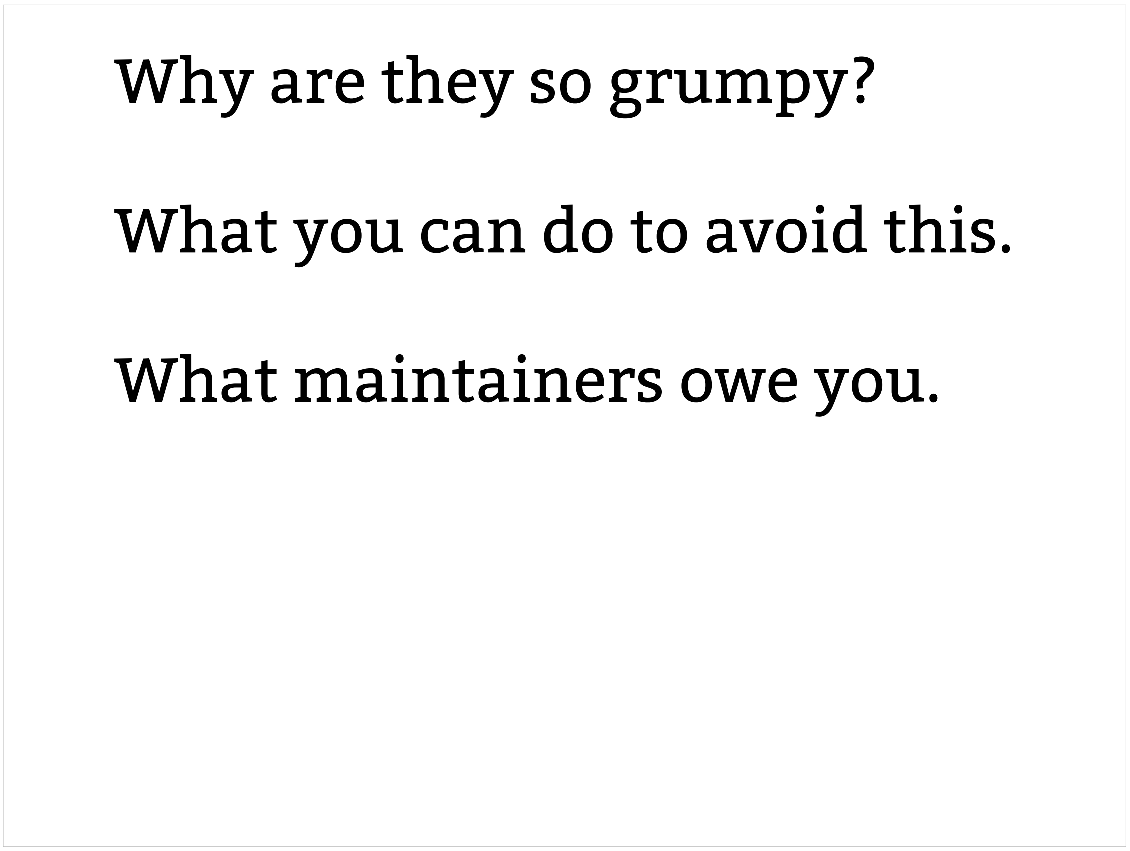
github.com/gregkh/presentation-maintainer

Linux Kernel Maintainers

Greg Kroah-Hartman
gregkh@linuxfoudation.org

When I first started writing this talk, it quickly turned into one big long rant. I ended up listing all of the different problems that I had with patches that people had sent me over the past few years.

While this would have been a very fun and cathartic talk for me, I figured that you all just watching me complain for 30 minutes wouldn't be the most intertaining thing, so I figured I would try to tone it down.


Why are they so grumpy?

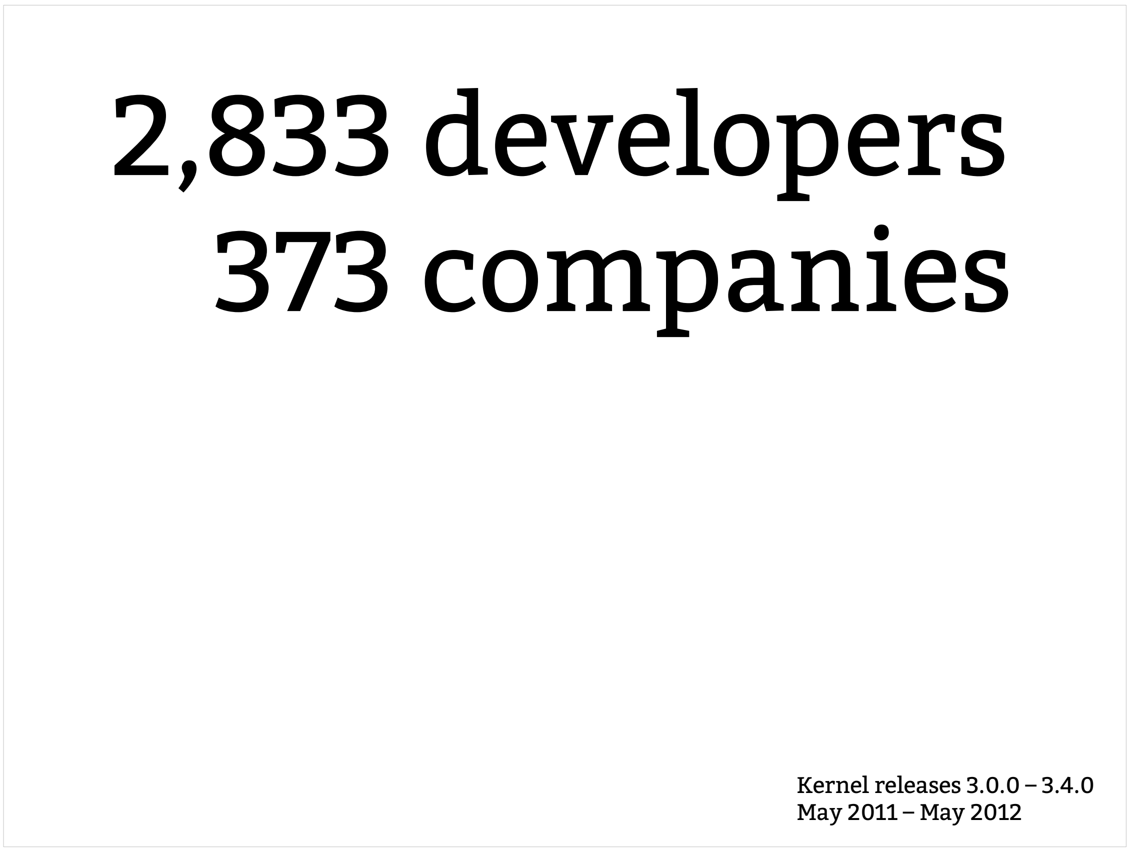
What you can do to avoid this.

What maintainers owe you.

So, let's talk about the main problem that people seem to have with Linux kernel maintainers, why are they so grumpy? Hopefully by the end of this talk, you will have an idea of why this always happens, and what you can do to avoid having that anger be directed at you.

Also, I'm going to cover what you should expect from a good kernel maintainer, so if you are a maintainer, here's something that developers can use to get back at you, and me, as I figure it's only fair.

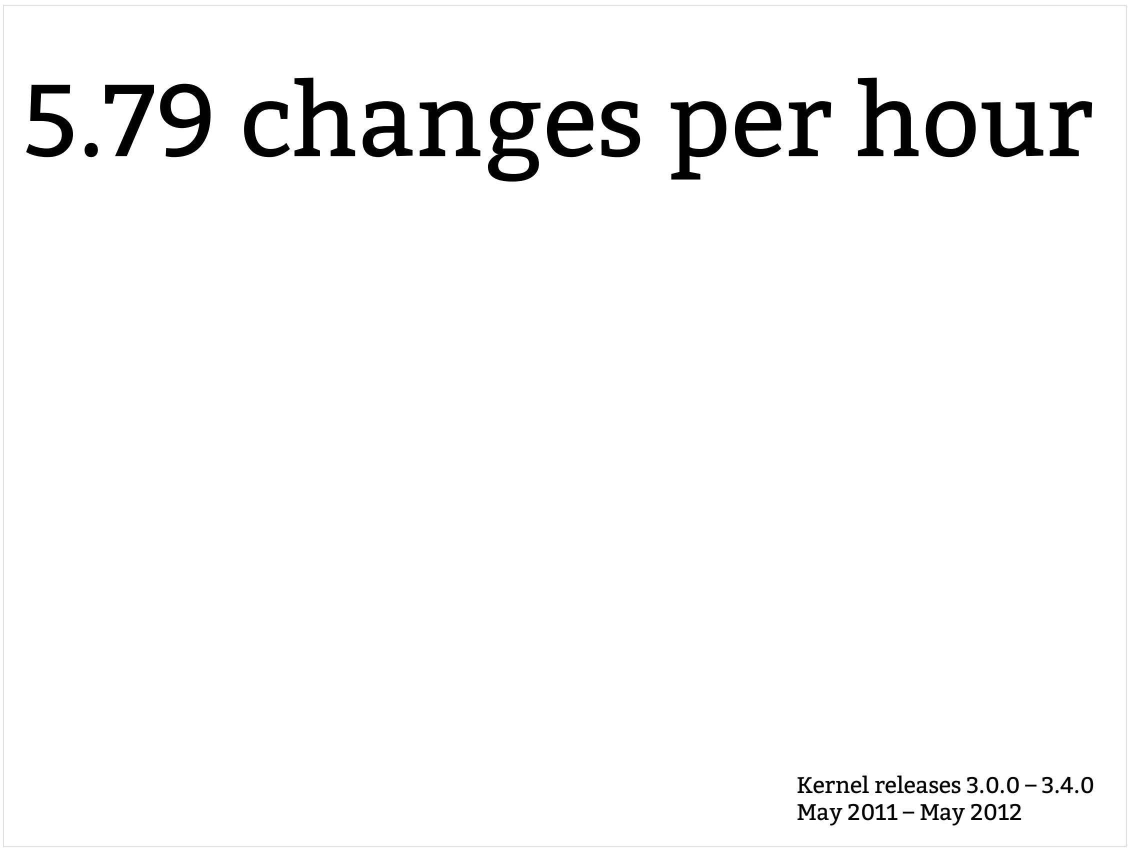
I am going to complain a lot in this talk. Please don't get the impression that I don't like doing this type of work. I love it. It's the best job in the world that I've ever had, and I can't think of anything that I would rather be doing.


2,833 developers
373 companies

Kernel releases 3.0.0 – 3.4.0
May 2011 – May 2012

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

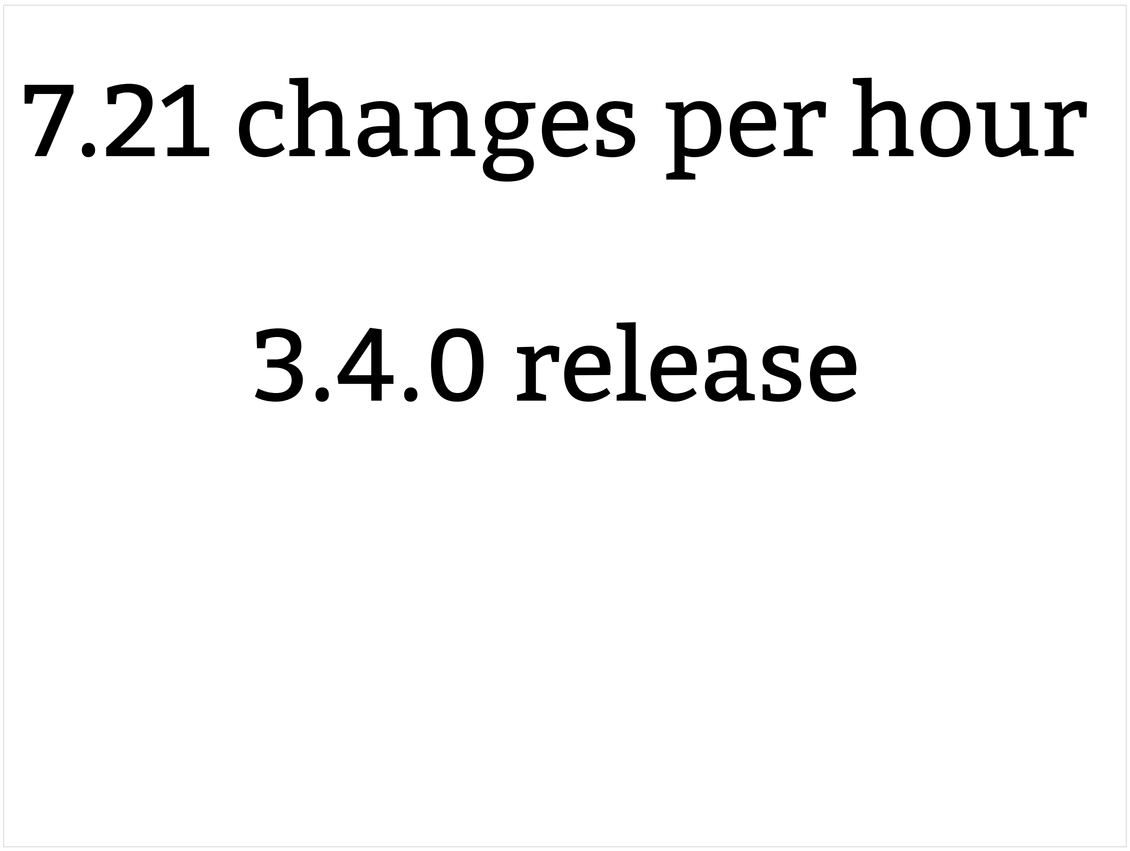
This is just the number of companies that we know about, there are more that we do not, and as the responses to our inquiries come in, this number will go up.



5.79 changes per hour

Kernel releases 3.0.0 – 3.4.0
May 2011 – May 2012

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?

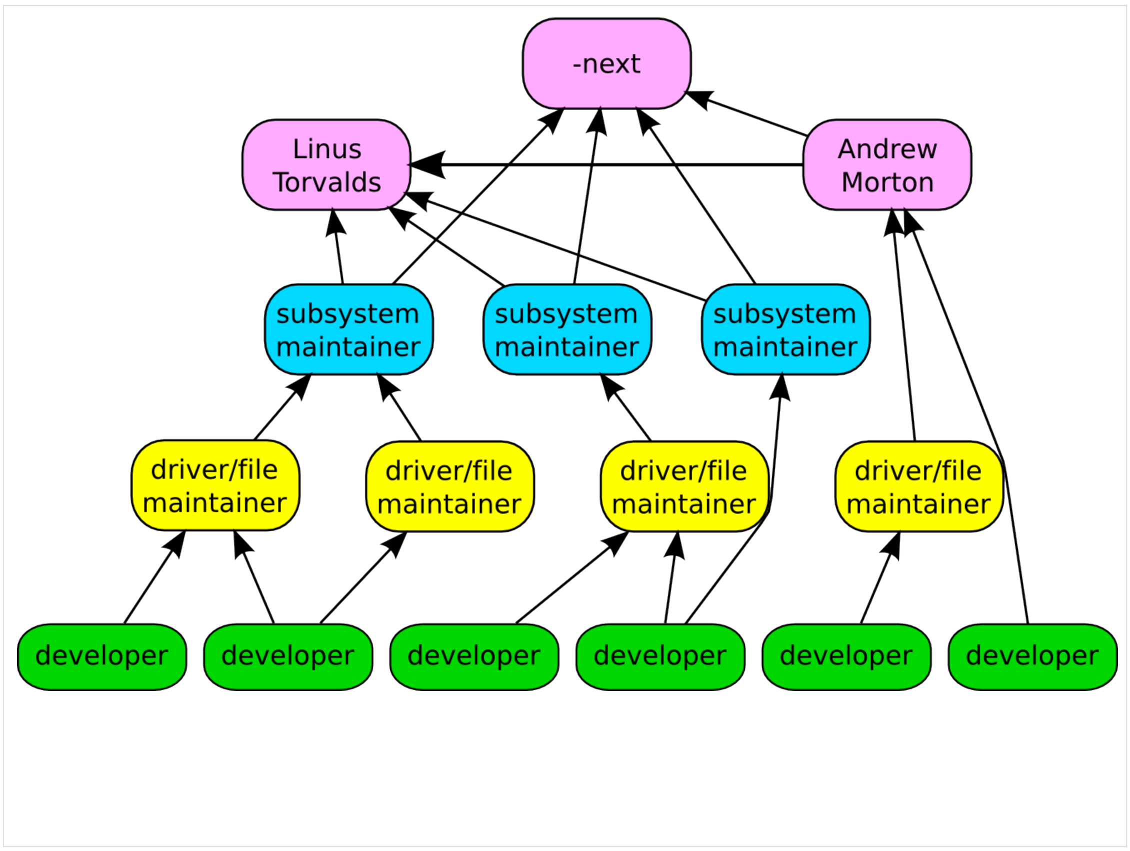


7.21 changes per hour

3.4.0 release

This past 3.4 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.

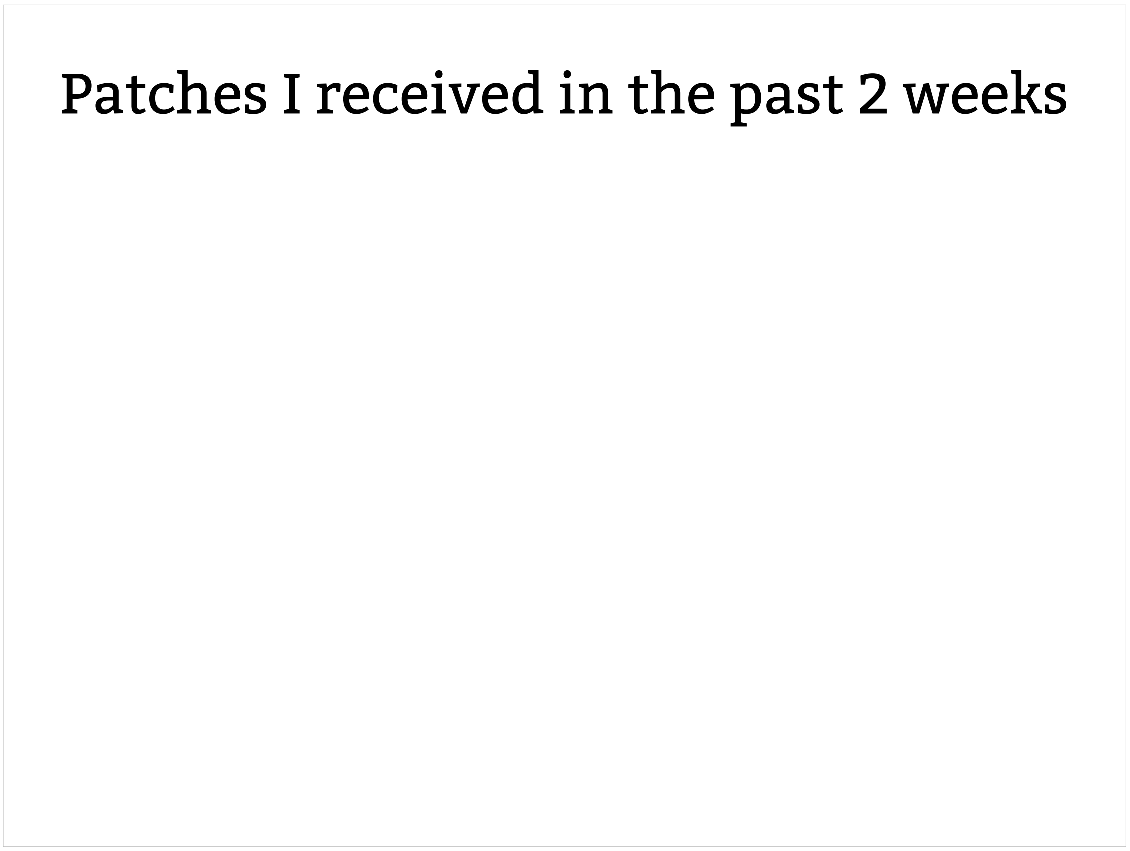
Now this is just the patches we accepted, not all of the patches that have been submitted, lots of patches are rejected, as anyone who has ever tried to submit a patch can attest to.



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 130 different subsystem maintainers at the moment.

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.

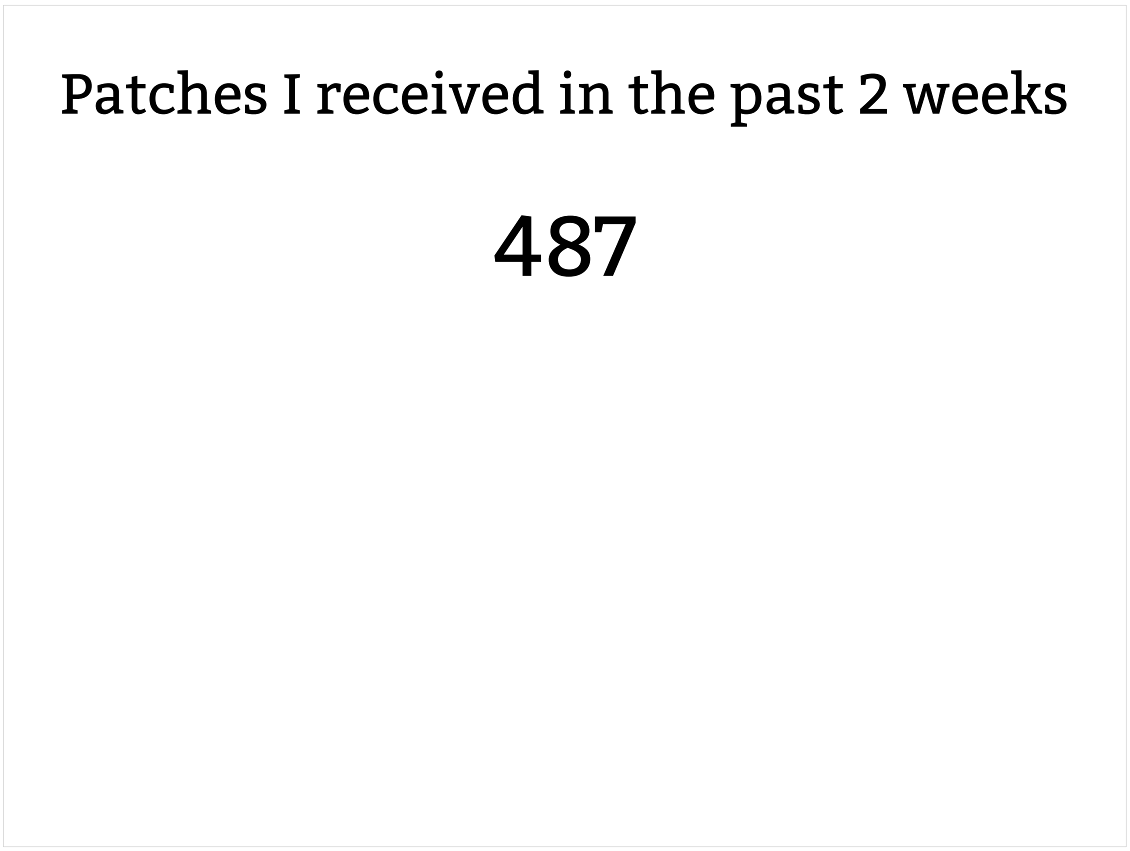


Patches I received in the past 2 weeks

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.

These past 2 weeks is the timeframe 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 the past 2 weeks

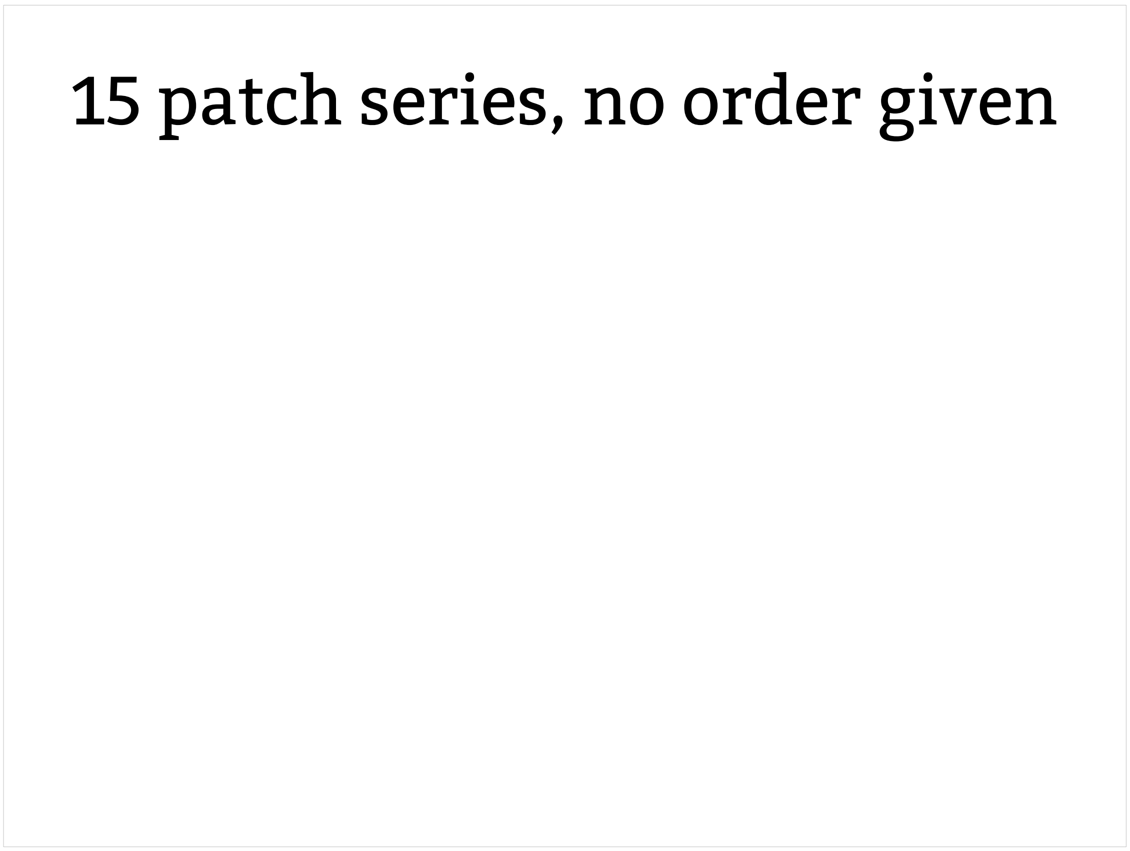
487

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.



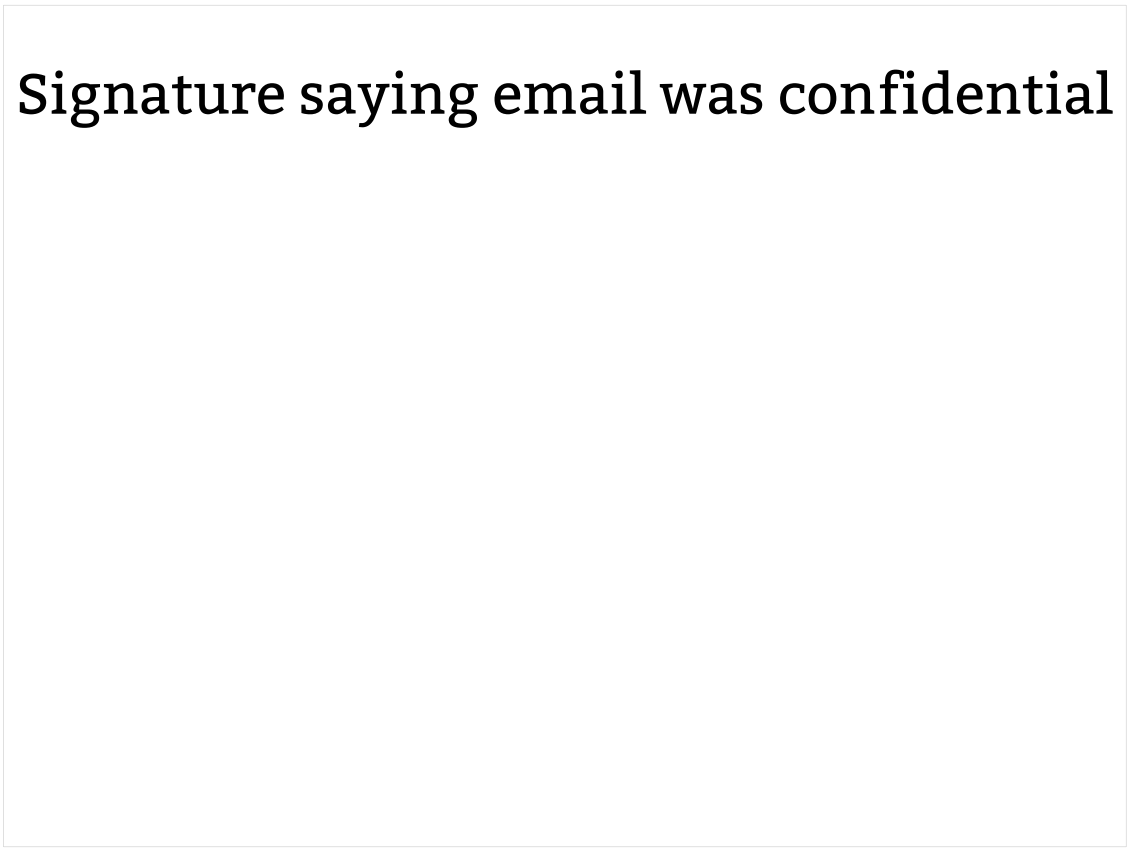
15 patch series, no order given

Am I supposed to guess?

Patches 1, 3-10

“Signed-off-by:” in signature

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



Signature saying email was confidential

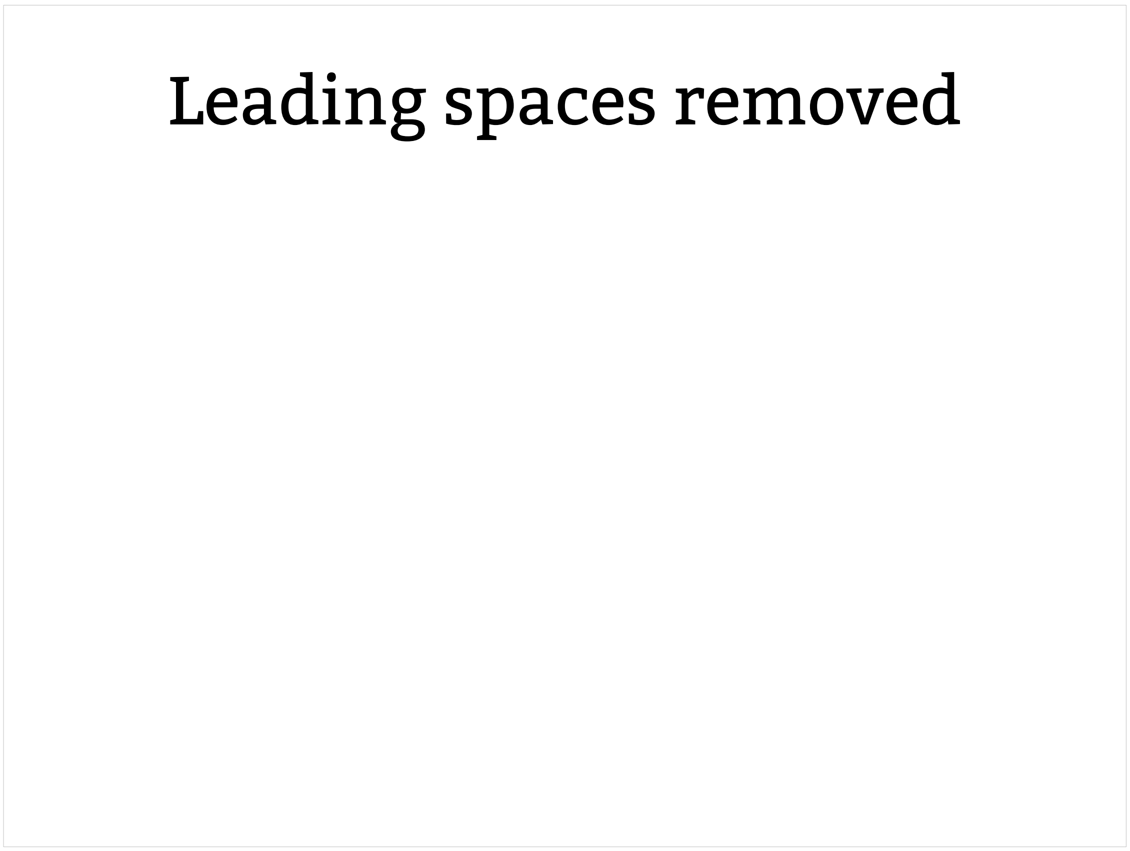
That kind of goes against how you are supposed to be sending Linux kernel patches out to the world.

Tabs were converted to spaces

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.



Leading spaces removed

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.

diff in non-unified format

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

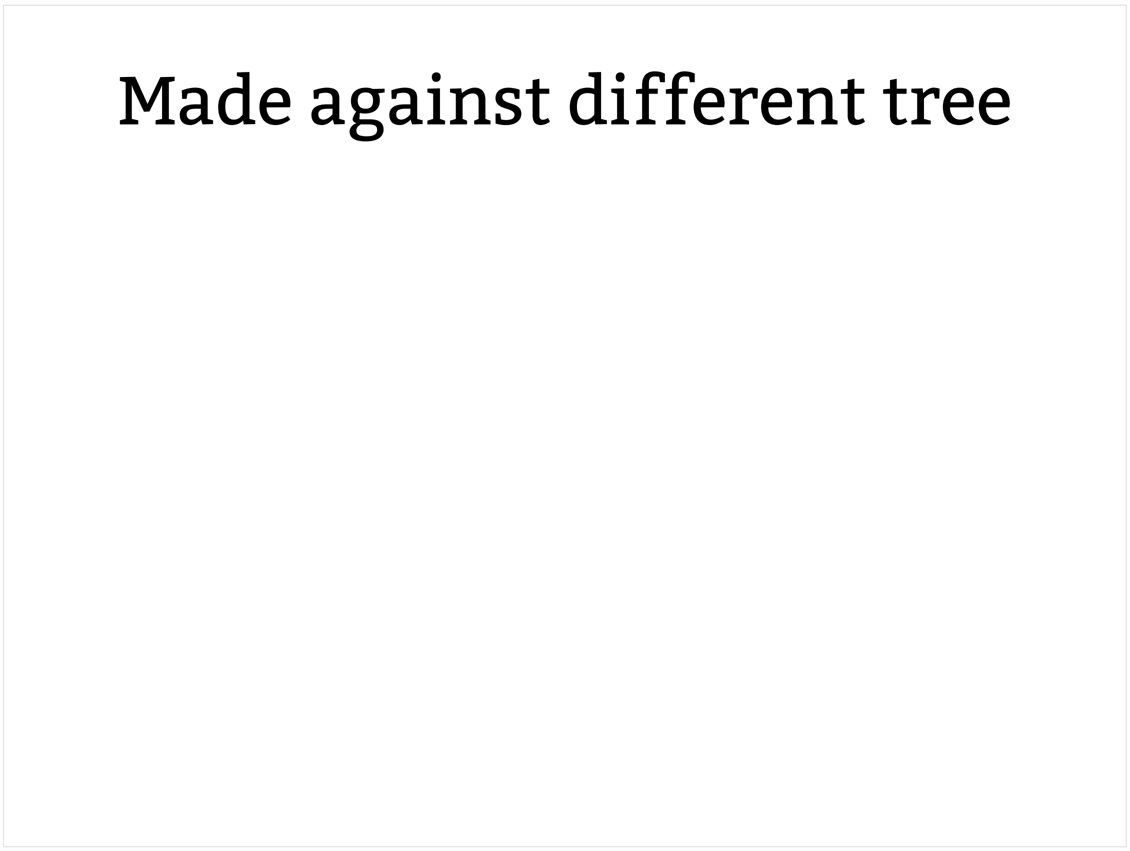
Patches need to be created in the root of the kernel source tree, as that's where I have to be in order to apply them properly.

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

Patch created in /usr/src/linux-2.6.32

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.

Wrong coding style

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

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)

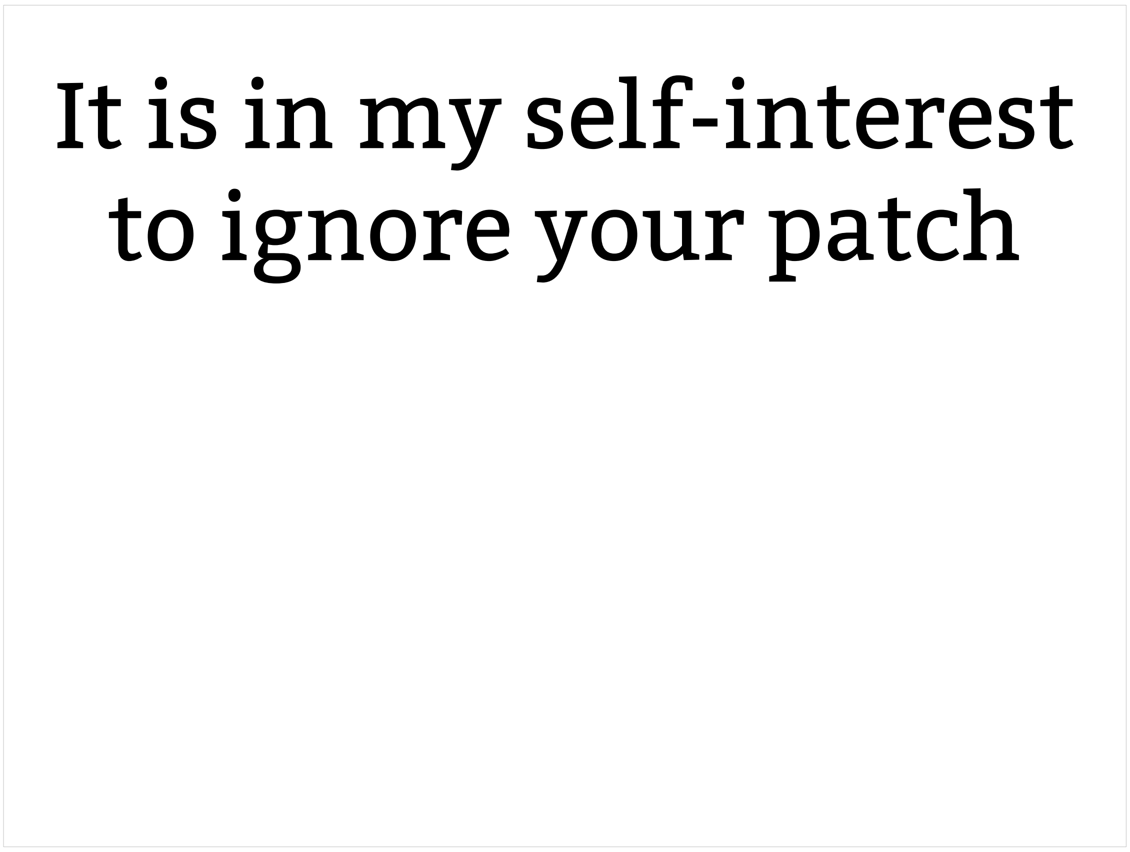
Obviously wrong kerneldoc

This was a calm two weeks

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 on a constant basis.

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.



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:

Proper coding style

scripts/checkpatch.pl clean

Sent to proper people and lists

Sent to proper people and lists

scripts/get_maintainer.pl

Proper Subject:

Proper changelog comment

Description of WHY it is needed

Small incremental change

“obviously” correct

Which tree it was made against

If multiple patches, state the order

Has to build properly

Make sure it works, if possible

Don't ignore review comments

Don't resend without saying why

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

Obligatory Penguin Picture

[redacted]