regressions: our workflows are the biggest enemy

Thorsten Leemhuis

[1. maintainers summit two days ago]

what was discussed and decided

a lot and nothing ;-)

hard to summarize, but was a good discussion

Linus is willing to ack
a few docs describing
what's expected from
developers and maintainers

Linus wants everything from current cycle fixed by -rc6

Linus wants regressions fixed quickly that hit versions deemed for end users

Linus thinks -next not that important for regression fixes

one or two days good idea for CIs!

[2. regzbot]

(my regression tracking bot)

halted development

a few months ago, to be precise

last big features not widely announced yet

new feature:

support for monitoring issue trackers from gitlab and github

new feature:

support for updating regzbot data without replying to the report

new feature:

a few quality of life improvements; more needed

missing

a few minor but kinda important things that would be really useful

like adding something to the tracking
while posting a fix for the problem

missing

subsystem specific views
into tracked regression

(reports & websites)

missing

better interactions with CIs

missing

a lot of other things people asked for

various reasons why development was stopped

[for the moment]

future: should get better

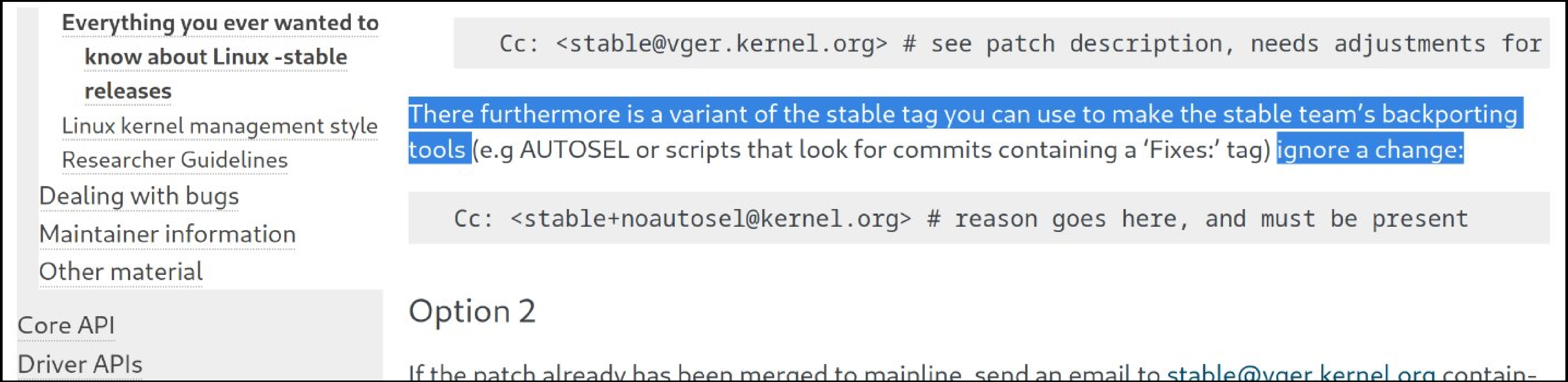
[3. why do regressions happen]

because kernel developers are human ;-)

because the kernel is mainly drivers and thus hard to test

mainline: because fixing regressions takes too long

[no silver bullet to solve this, as its caused by various small issues that on its own look negligible] stable: because stable team has no access to tests subsystem maintainers usually run stable: stable team unable to reliably distinct fixes worth backporting from those it should ignore



```
Cc: <stable@vger.kernel.org> # see patch description, needs adjustments for
```

There furthermore is a variant of the stable tag you can use to make the stable team's backporting tools (e.g AUTOSEL or scripts that look for commits containing a 'Fixes:' tag) ignore a change:

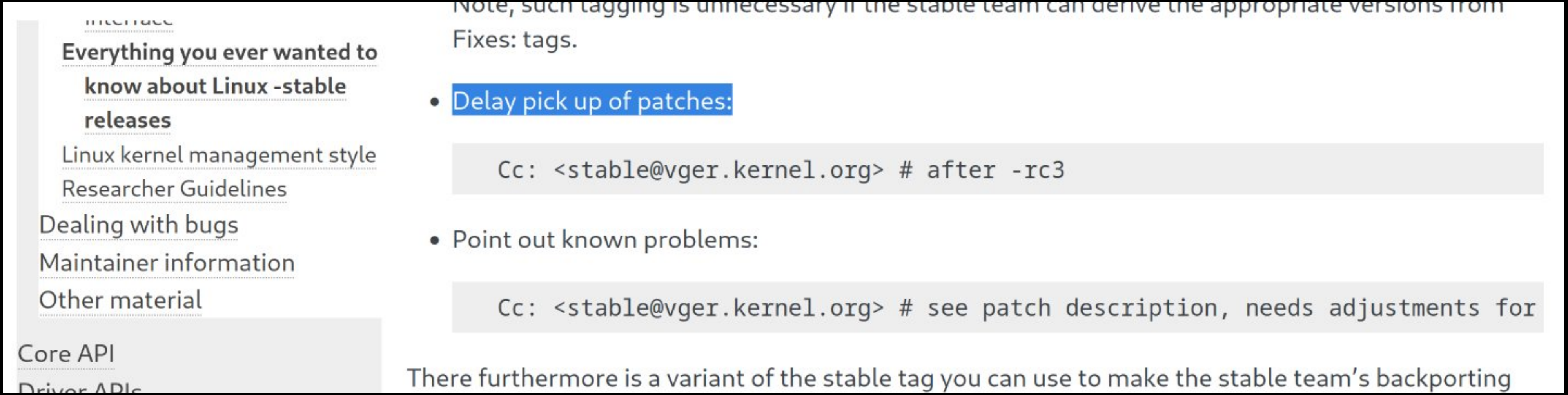
```
Cc: <stable+noautosel@kernel.org> # reason goes here, and must be present
```

Option 2

If the patch already has been merged to mainline, send an email to stable@vger.kernel.org contain-

https://docs.kernel.org/process/stable-kernel-rules.html

stable: stable team unable
to distinct fixes worth prioritizing
from those better delayed



Fixes: tags.

- Delay pick up of patches:

```
Cc: <stable@vger.kernel.org> # after -rc3
```

- Point out known problems:

```
Cc: <stable@vger.kernel.org> # see patch description, needs adjustments for
```

There furthermore is a variant of the stable tag you can use to make the stable team's backporting

https://docs.kernel.org/process/stable-kernel-rules.html stable: because changes are backported rather quicky stable team has no simple way to detect if fixes are part of a patch-set stable team has no way to detect if fixes implicitly depend on changes mainlined earlier

[4. workflow problems]

most kernel developers
are doing a great job!

most kernel maintainers
are even doing a hell of a job!

many many thx to everyone for their upstream work!

do not want to criticize anybody's work in this talk

I want to point out
issues with *our process*

because no change or
not even discussing it properly
due to fear of downsides
leads to downfalls of empires

"workflows are the biggest enemy":

"workflows are the biggest enemy":

there is no workflow for handling regressions 🥴

Everybody works with different interpretations of LKML mails from Linus

Everybody works with different interpretations of LKML mails from Linus

Everybody works with different interpretations of varying subsets of LKML mails from Linus

Everybody works with different interpretations of varying subsets of LKML mails from Linus, mixed with a combination of rules

Everybody works with different interpretations of varying subsets of LKML mails from Linus, mixed with a combination of rules either global or subsystem-specific

Everybody works with different interpretations of varying subsets of LKML mails from Linus, mixed with a combination of rules either global or subsystem-specific, official or unofficial

Everybody works with different interpretations of varying subsets of LKML mails from Linus, mixed with a combination of rules either global or subsystem-specific, official or unofficial, written or unwritten

Everybody works with different interpretations of varying subsets of LKML mails from Linus, mixed with a combination of rules either global or subsystem-specific, official or unofficial, written or unwritten, some of which are outdated

Everybody works with different interpretations of varying subsets of LKML mails from Linus, mixed with a combination of rules either global or subsystem-specific, official or unofficial, written or unwritten, some of which are outdated or contradicting each other

Everybody works with different interpretations of varying subsets of LKML mails from Linus, mixed with a combination of rules either global or subsystem-specific, official or unofficial, written or unwritten, some of which are outdated or contradicting each other – while being free to not care at all if a regressions made it into a release deemed for end users.

let me illustrate the problem with the audience

A patch is merged for 6.10-rc1;

A patch is merged for 6.10-rc1;
two days after 6.10-rc1 is out someone reports it causes a regression;

A patch is merged for 6.10-rc1;
two days after 6.10-rc1 is out someone reports it causes a regression;
a straight-forward fix is posted, reviewed, and merged to a subsystem tree on the Friday before 6.10-rc4 comes out.

A patch is merged for 6.10-rc1;
two days after 6.10-rc1 is out someone reports it causes a regression;
a straight-forward fix is posted, reviewed, and merged to a subsystem tree on the Friday before 6.10-rc4 comes out.

Should it be send to Linus the following Friday for mainlining into 6.10-rc5?

A patch is merged for **6.9-rc1**;
two days after 6.10-rc1 is out someone reports it causes a regression;
a straight-forward fix is posted, reviewed, and merged to a subsystem tree on the Friday before 6.10-rc4 comes out.

Should it be send to Linus the following Friday for mainlining into 6.10-rc5?

A patch is merged for 6.9-rc1;
two days after 6.10-rc1 is out someone reports it causes a regression;
a straight-forward fix is posted, reviewed, and merged to a subsystem tree on the Friday before **6.10-rc6** comes out.

Should it be send to Linus the following Friday for mainlining into ***6.10-rc7****?*

A patch is merged for 6.9-rc1;
two days after 6.10-rc1 is out someone reports it causes a regression;
a straight-forward fix is posted, reviewed, and merged to a subsystem tree on the Friday before **6.10-rc7** comes out.

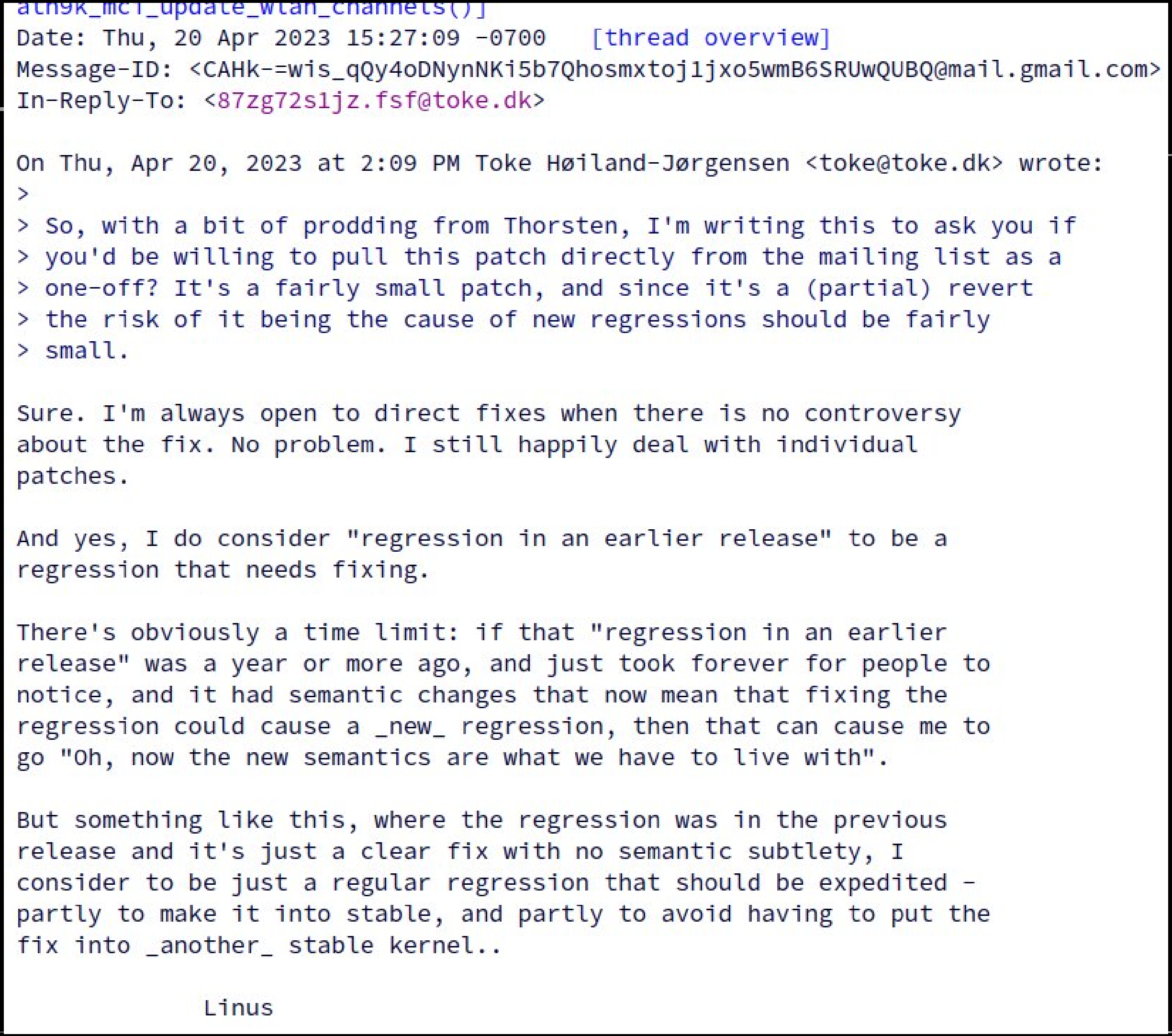
*Should it be send to Linus the following Friday for mainlining into **6.10**?*

A patch is merged for **6.6-rc1**;
two days after 6.10-rc1 is out someone reports it causes a regression;
a straight-forward fix is posted, reviewed, and merged to a subsystem tree on the Friday before 6.10-rc7 comes out.

Should it be send to Linus the following Friday for mainlining into 6.10?

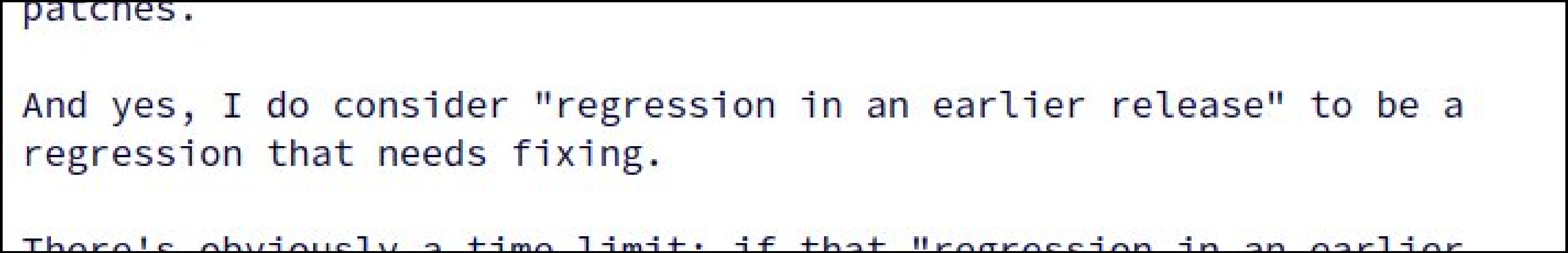
A patch is merged for **6.1-rc1**;
two days after 6.10-rc1 is out someone reports it causes a regression;
a straight-forward fix is posted, reviewed, and merged to a subsystem tree on the Friday before 6.10-rc7 comes out.

Should it be send to Linus the following Friday for mainlining into 6.10?



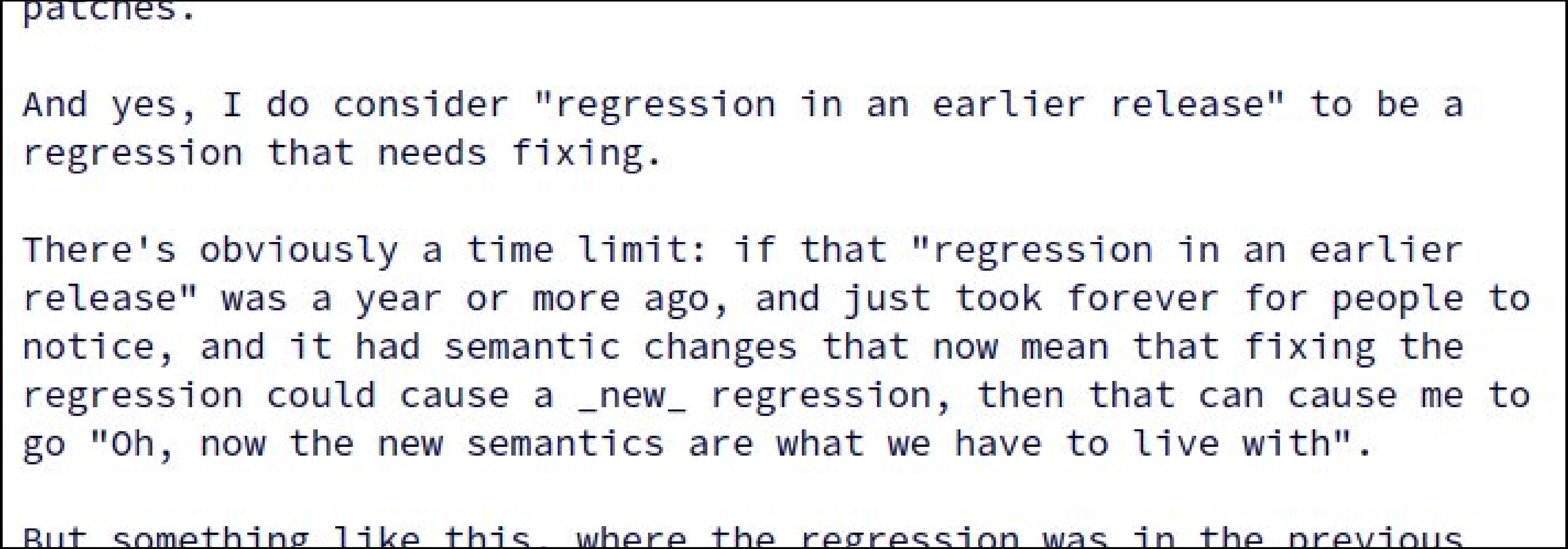
```
ath9k_mci_update_wlan_channels()]
Date: Thu, 20 Apr 2023 15:27:09 -0700   [thread overview]
Message-ID: <CAHk-=wis_qQy4oDNynNKi5b7Qhosmxtoj1jxo5wmB6SRUwQUBQ@mail.gmail.com>
In-Reply-To: <87zg72s1jz.fsf@toke.dk>

On Thu, Apr 20, 2023 at 2:09 PM Toke Høiland-Jørgensen <toke@toke.dk> wrote:
>
> So, with a bit of prodding from Thorsten, I'm writing this to ask you if
> you'd be willing to pull this patch directly from the mailing list as a
> one-off? It's a fairly small patch, and since it's a (partial) revert
> the risk of it being the cause of new regressions should be fairly
> small.

Sure. I'm always open to direct fixes when there is no controversy
about the fix. No problem. I still happily deal with individual
patches.

And yes, I do consider "regression in an earlier release" to be a
regression that needs fixing.

There's obviously a time limit: if that "regression in an earlier
release" was a year or more ago, and just took forever for people to
notice, and it had semantic changes that now mean that fixing the
regression could cause a _new_ regression, then that can cause me to
go "Oh, now the new semantics are what we have to live with".

But something like this, where the regression was in the previous
release and it's just a clear fix with no semantic subtlety, I
consider to be just a regular regression that should be expedited -
partly to make it into stable, and partly to avoid having to put the
fix into _another_ stable kernel..

             Linus
```

https://lore.kernel.org/all/CAHk-=wis_qQy4oDNynNKi5b7Qhosmxtoj1jxo5wmB6SRUwQUBQ@mail.gmail.com/



```
patches.

And yes, I do consider "regression in an earlier release" to be a
regression that needs fixing.

There's obviously a time limit: if that "regression in an earlier
```

https://lore.kernel.org/all/CAHk-=wis_qQy4oDNynNKi5b7Qhosmxtoj1jxo5wmB6SRUwQUBQ@mail.gmail.com/



patches.

And yes, I do consider "regression in an earlier release" to be a regression that needs fixing.

There's obviously a time limit: if that "regression in an earlier release" was a year or more ago, and just took forever for people to notice, and it had semantic changes that now mean that fixing the regression could cause a _new_ regression, then that can cause me to go "Oh, now the new semantics are what we have to live with".

But something like this, where the regression was in the previous

https://lore.kernel.org/all/CAHk-=wis_qQy4oDNynNKi5b7Qhosmxtoj1jxo5wmB6SRUwQUBQ@mail.gmail.com/

A lot of regressions despite the lack of agreed on guidelines are handled quite well many regression are fixed within one, two, or three weeks 😃

others take three months 😟

others take three months 😟
and/or are never fixed in the series
in which they were introduced 😠

A lot of regressions despite the lack of agreed on guidelines are handled quite well

A lot of regressions despite the
lack of agreed on guidelines
are handled quite well –
but many leave a lot to wish for, too.

A patch is merged for 6.10-rc1;

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;

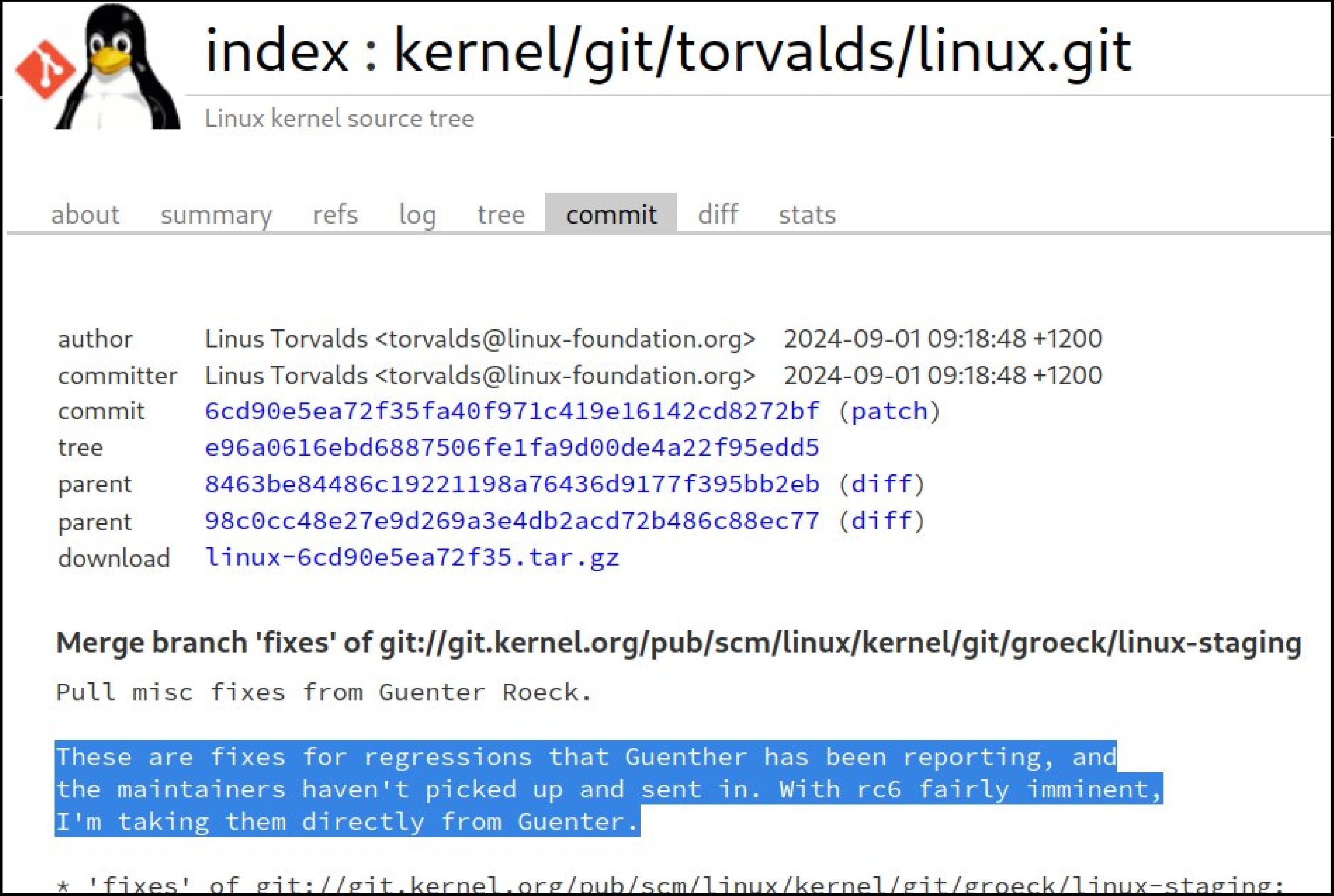
A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc2.

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Monday morning;
during Monday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc3.

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Monday morning;
during Friday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc4.

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Monday morning;
during Friday it then is committed to a subsystem tree;
it makes it to -next a day later;
after one and a half weeks is then send to Linus;
Linus picks it up for 6.10-rc5.

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Monday morning;
during Friday it then is committed to a subsystem tree;
it makes it to -next a day later;
after two and a half weeks is then send to Linus;
Linus picks it up for 6.10-rc6.



index : kernel/git/torvalds/linux.git

Linux kernel source tree

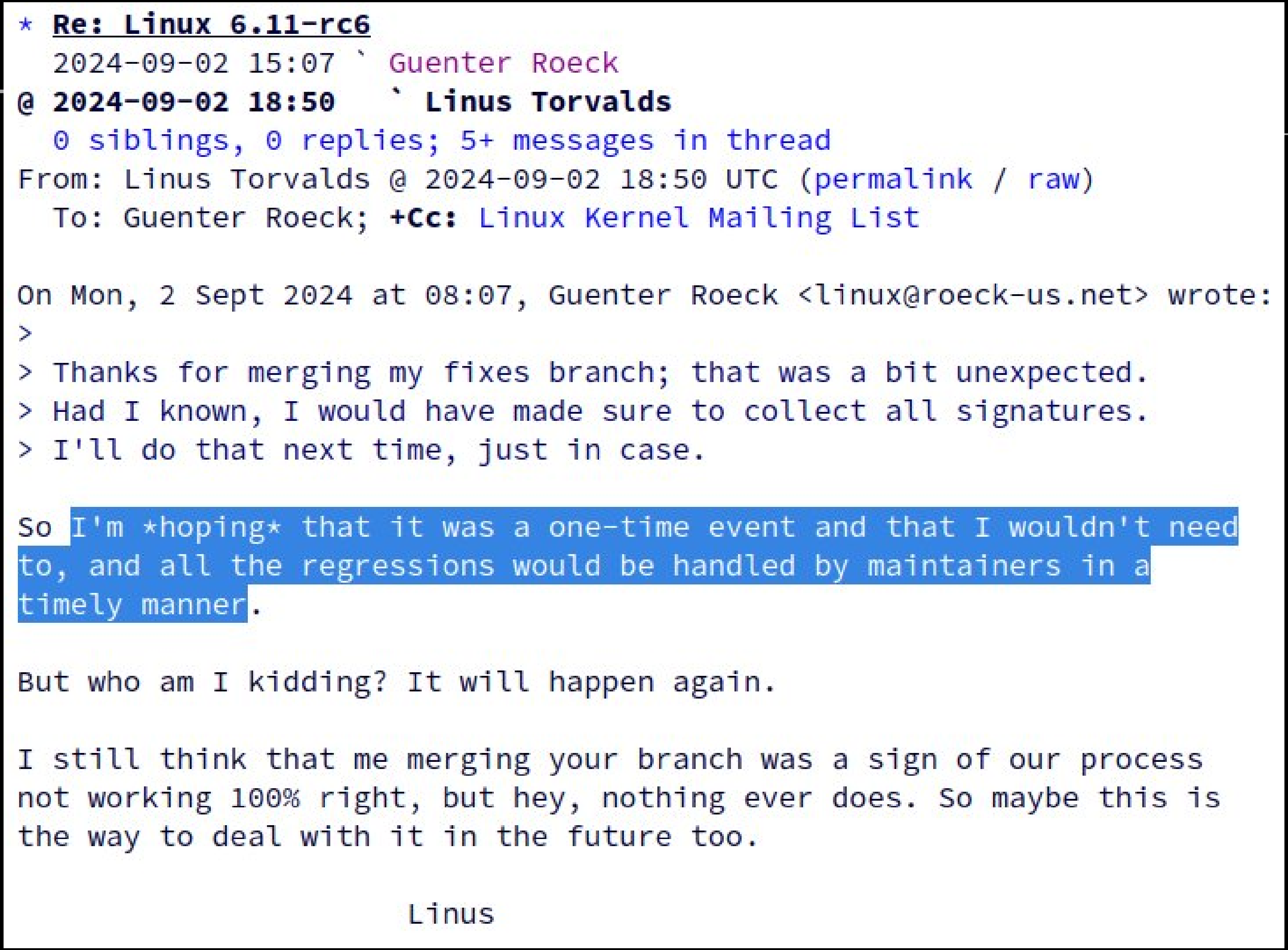
about summary refs log tree commit diff stats

author	Linus Torvalds <torvalds@linux-foundation.org>	2024-09-01 09:18:48 +1200
committer	Linus Torvalds <torvalds@linux-foundation.org>	2024-09-01 09:18:48 +1200
commit	6cd90e5ea72f35fa40f971c419e16142cd8272bf (patch)	
tree	e96a0616ebd6887506fe1fa9d00de4a22f95edd5	
parent	8463be84486c19221198a76436d9177f395bb2eb (diff)	
parent	98c0cc48e27e9d269a3e4db2acd72b486c88ec77 (diff)	
download	linux-6cd90e5ea72f35.tar.gz	

Merge branch 'fixes' of git://git.kernel.org/pub/scm/linux/kernel/git/groeck/linux-staging

```
Pull misc fixes from Guenter Roeck.

These are fixes for regressions that Guenther has been reporting, and
the maintainers haven't picked up and sent in. With rc6 fairly imminent,
I'm taking them directly from Guenter.

* 'fixes' of git://git.kernel.org/pub/scm/linux/kernel/git/groeck/linux-staging:
```

https://git.kernel.org/torvalds/c/6cd90e5ea72f35



```
* Re: Linux 6.11-rc6
  2024-09-02 15:07 ` Guenter Roeck
@ 2024-09-02 18:50   ` Linus Torvalds
  0 siblings, 0 replies; 5+ messages in thread
From: Linus Torvalds @ 2024-09-02 18:50 UTC (permalink / raw)
  To: Guenter Roeck; +Cc: Linux Kernel Mailing List

On Mon, 2 Sept 2024 at 08:07, Guenter Roeck <linux@roeck-us.net> wrote:
>
> Thanks for merging my fixes branch; that was a bit unexpected.
> Had I known, I would have made sure to collect all signatures.
> I'll do that next time, just in case.

So I'm *hoping* that it was a one-time event and that I wouldn't need
to, and all the regressions would be handled by maintainers in a
timely manner.

But who am I kidding? It will happen again.

I still think that me merging your branch was a sign of our process
not working 100% right, but hey, nothing ever does. So maybe this is
the way to deal with it in the future too.

                      Linus
```

https://lore.kernel.org/all/CAHk-%3Dwinf5CvEvEGMUQmEYWFtp9b6YE0%2BSwhVTB89OpHRS_ZEA@mail.gmail.com/

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Monday morning;
during Friday it then is committed to a subsystem tree;
it makes it to -next a day later;
after two and a half weeks is then send to Linus;
Linus picks it up for 6.10-rc6.

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Monday morning;
during Friday it then is committed to a subsystem tree;
it makes it to -next a day later;
after two and a half weeks is then send to Linus;
Linus picks it up for 6.10-rc6.

workflow problem: some subsystems take a long time to review, commit, or mainline fixes

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Monday morning;
during Friday it then is committed to a subsystem tree;
it makes it to -next a day later;
after two and a half weeks is then send to Linus;
Linus picks it up for 6.10-rc6.

workflow problem: some subsystems lack workforce to review, commit, or mainline faster

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Monday morning;
during Friday it then is committed to a subsystem tree;
it makes it to -next a day later;
after two and a half weeks is then send to Linus;
Linus picks it up for 6.10-rc6.

workflow problem: some sub-subsystems see no urgency

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Monday morning;
during Friday it then is committed to a subsystem tree;
it makes it to -next a day later;
after two and a half weeks is then send to Linus;
Linus picks it up for 6.10-rc6.

workflow problem: submitted regression fixes often look like any other patch

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Monday morning;
during Friday it then is committed to a subsystem tree;
it makes it to -next a day later;
after two and a half weeks is then send to Linus;
Linus picks it up for 6.10-rc6.

workflow problem: some maintainers don't want to send Linus or their upstream a PR with just one fix

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Monday morning;
during Friday it then is committed to a subsystem tree;
it makes it to -next a day later;
after two and a half weeks is then send to Linus;
Linus picks it up for 6.10-rc6.

workflow problem: maintainers send PRs shortly after a new -rc

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Monday morning;
during Friday it then is committed to a subsystem tree;
it makes it to -next a day later;
after two and a half weeks is then send to Linus;
Linus picks it up for 6.10-rc6.

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Monday morning;
during Friday it then is committed to a subsystem tree;
it makes it to -next a day later;
after two and a half weeks is then send to Linus;
Linus picks it up for 6.10-rc6.

workflow problem: different assumptions on how long patches should be in -next

A patch is merged for 6.10-rc1;
on Tuesday morning after 6.10-rc7 someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
should this be send to Linus for inclusion in 6.10
right before its release?

A patch is merged for 6.10-rc1;
on Tuesday morning after 6.10-rc7 someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
should this be send to Linus for inclusion in 6.10
right before its release?

workflow problem: some devs/maintainers are too careful when it comes to mainlining last minute regression fixes

A patch is merged for 6.10-rc1;
on Tuesday morning after 6.10-rc7 someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Friday it then is committed to a subsystem tree;
it thus never makes it to -next;
should this be send to Linus for inclusion in 6.10
right before its release?

A patch is merged for 6.10-rc1;
on Tuesday morning after 6.10-rc7 someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Friday it then is committed to a subsystem tree;
it thus never makes it to -next;
should this be send to Linus for inclusion in 6.10
right before its release?

workflow problem: developers are unsure if sending fixes to Linus that never have been in -next

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc2.

[reset to the one week example]

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix,
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc2.

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix,
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc2.

workflow problem: reporting

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc2.

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc2.

***usually not* a problem: developing a fix** 😃

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after five weeks of debugging, tests et. al. no fix is found and
developers decide to apply a revert;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc7.

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after five weeks of debugging, tests et. al. no fix is found and
developers decide to apply a revert;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc7.

workflow problem: some developers try hard to avoid reverts

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc2.

[reset to the one week example]

A patch is merged for 6.9-rc1;
on Tuesday morning after the 6.10-rc6 release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it then is send to Linus during the next merge window;
Linus picks it up for 6.11-rc1.

A patch is merged for 6.6-rc1;
on Tuesday morning after the 6.10-rc6 release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it then is send to Linus during the next merge window;
Linus picks it up for 6.11-rc1.

A patch is merged for 6.6-rc1;
on Tuesday morning after the 6.10-rc6 release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it then is send to Linus during the next merge window;
Linus picks it up for 6.11-rc1.

workflow problem: some devs unsure where to queue fixes for older regressions: for the current or next cycle?

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
after debugging a straight-forward fix is posted on
Wednesday morning and tested within hours;
it then is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc2.

[reset to the one week example]

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc2.

[and compress content and formatting it a little]

A patch is merged for 6.10-rc1;
on Tuesday morning after its release someone reports
it causes a regression a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc2.

A patch is merged after 6.9-rc7;
on Tuesday morning after the 6.9 release someone reports
it causes a regression a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it within a day or two is then send to Linus;
Linus picks it up for 6.10-rc2.

A patch is merged after 6.9-rc7;
on Tuesday morning after the 6.9 release someone reports
it causes a regression a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after six weeks in -next is then send to Linus;
Linus picks it up for 6.10-rc6.

A patch is merged after 6.9-rc7;
on Tuesday morning after the 6.9 release someone reports
it causes a regression a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after six weeks in -next is then send to Linus;
Linus picks it up for 6.10-rc6.

workflow problem: no elevated handling for regressions that recently made it into a release deemed for end users

A patch is merged after 6.9-rc7;
on Tuesday morning after the 6.9 release someone reports
it causes a regression a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after nine weeks in -next is then send to Linus;
Linus picks it up for 6.11-rc1.

workflow problem: no elevated handling for regressions that recently made it into a release deemed for end users

A patch is merged after 6.9-rc7;
on Tuesday morning after the 6.9 release someone reports
it causes a regression a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after nine weeks in -next is then send to Linus;
Linus picks it up for 6.11-rc1.
The fix contains a 'Fixes:…' tag, but no 'CC: <stable@…'; it's never
backported to 6.9.y or 6.10.y and reaches users only with 6.11.

A patch is merged after 6.9-rc7;
on Tuesday morning after the 6.9 release someone reports
it causes a regression a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after nine weeks in -next is then send to Linus;
Linus picks it up for 6.11-rc1.
The fix contains a 'Fixes:…' tag, but no 'CC: <stable@…'; it's never backported to 6.9.y or 6.10.y and reaches users only with 6.11.

workflow problem: fixes tags are sometimes are missing or wrong

A patch is merged after 6.9-rc7;
on Tuesday morning after the 6.9 release someone reports
it causes a regression a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after nine weeks in -next is then send to Linus;
Linus picks it up for 6.11-rc1.
The fix contains a 'Fixes:…' tag, but not 'CC: <stable@…'; it's never backported to 6.9.y or 6.10.y and reaches users only with 6.11.

related workflow problem: a lot of developers assume Fixes: tags suffice to initiate backporting – which they don't!

A patch is merged after 6.9-rc7;
on Tuesday morning after the 6.9 release someone reports
it causes a regression a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after nine weeks in -next is then send to Linus;
Linus picks it up for 6.11-rc1.
The fix contains a 'Fixes:…' tag, but not 'CC: <stable@…'; it's never backported to 6.9.y or 6.10.y and reaches users only with 6.11.

workflow problem: some subsystems opted-out of backporting commits with 'Fixes:' tag

A patch is merged after 6.9-rc7;
on Tuesday morning after the 6.9 release someone reports
it causes a regression a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after nine weeks in -next is then send to Linus;
Linus picks it up for 6.11-rc1.
The fix contains a 'Fixes:…' tag, but not 'CC: <stable@…'; it's never backported to 6.9.y or 6.10.y and reaches users only with 6.11.

related workflow problem: participating in stable(*) kernel maintenance is entirely optional for mainline developers

(*) this here and later means longterm aka LTS kernels as well

[everything up until the previous slide was about mainline]

[everything up until the previous slide was about mainline]

We just entered stable/longterm territory!

A patch is merged after 6.9-rc7; on Tuesday morning
after the 6.10-rc1 release someone reports
it causes a regression in 6.9.2 a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after nine weeks in -next is then send to Linus;
Linus picks it up for 6.11-rc1.
The fix contains a 'Fixes:…' tag, but not 'CC: <stable@…'; it's never backported to 6.9.y or 6.10.y and reaches users only with 6.11.

A patch is merged after 6.9-rc7; on Tuesday morning
after the 6.10-rc1 release someone reports
it causes a regression in 6.9.2 a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after nine weeks in -next is then send to Linus;
Linus picks it up for 6.11-rc1.
The fix contains a 'Fixes:…' tag, but not 'CC: <stable@…'; it's never backported to 6.9.y or 6.10.y and reaches users only with 6.11.

workflow problem: people report such bugs to the stable list

A patch is merged after 6.9-rc7; on Tuesday morning
after the 6.10-rc1 release someone reports
it causes a regression in 6.9.2 a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after nine weeks in -next is then send to Linus;
Linus picks it up for 6.11-rc1.
The fix contains a 'Fixes:…' tag, but not 'CC: <stable@…'; it's never
backported to 6.9.y or 6.10.y and reaches users only with 6.11.

workflow problem: mainline devs might not care because they might suspect its a bug stable bug introduced in 6.9.1 or 6.9.2

A patch is merged after 6.9-rc7; on Tuesday morning
after the 6.10-rc1 release someone reports
it causes a regression in 6.9.2 a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after nine weeks in -next is then send to Linus;
Linus picks it up for 6.11-rc1.
The fix contains a 'Fixes:…' tag, but not 'CC: <stable@…'; it's never
backported to 6.9.y or 6.10.y and reaches users only with 6.11.

**workflow problem: mainline developers not even in
this case are obliged to set a stable tag**

A patch is merged after 6.9-rc7; on Tuesday morning
after the 6.10-rc1 release someone reports
it causes a regression in 6.9.2 a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after nine weeks in -next is then send to Linus;
Linus picks it up for 6.11-rc1.
The fix contains a 'Fixes:…' tag, but not 'CC: <stable@…'; it's never
backported to 6.9.y or 6.10.y and reaches users only with 6.11.

workflow problem: mainline developers assume
a Fixes: tag is enough

A patch is merged after 6.10-rc1; on Wednesday it is
backported to 6.9.3 and right afterwards someone reports
it causes a regression in mainline and stable a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after five weeks in -next is then send to Linus;
Linus picks it up for 6.10-rc7.
The fix contains a 'Fixes:…' tag, but not 'CC: <stable@…';
it a few days later is backported to 6.9.y.

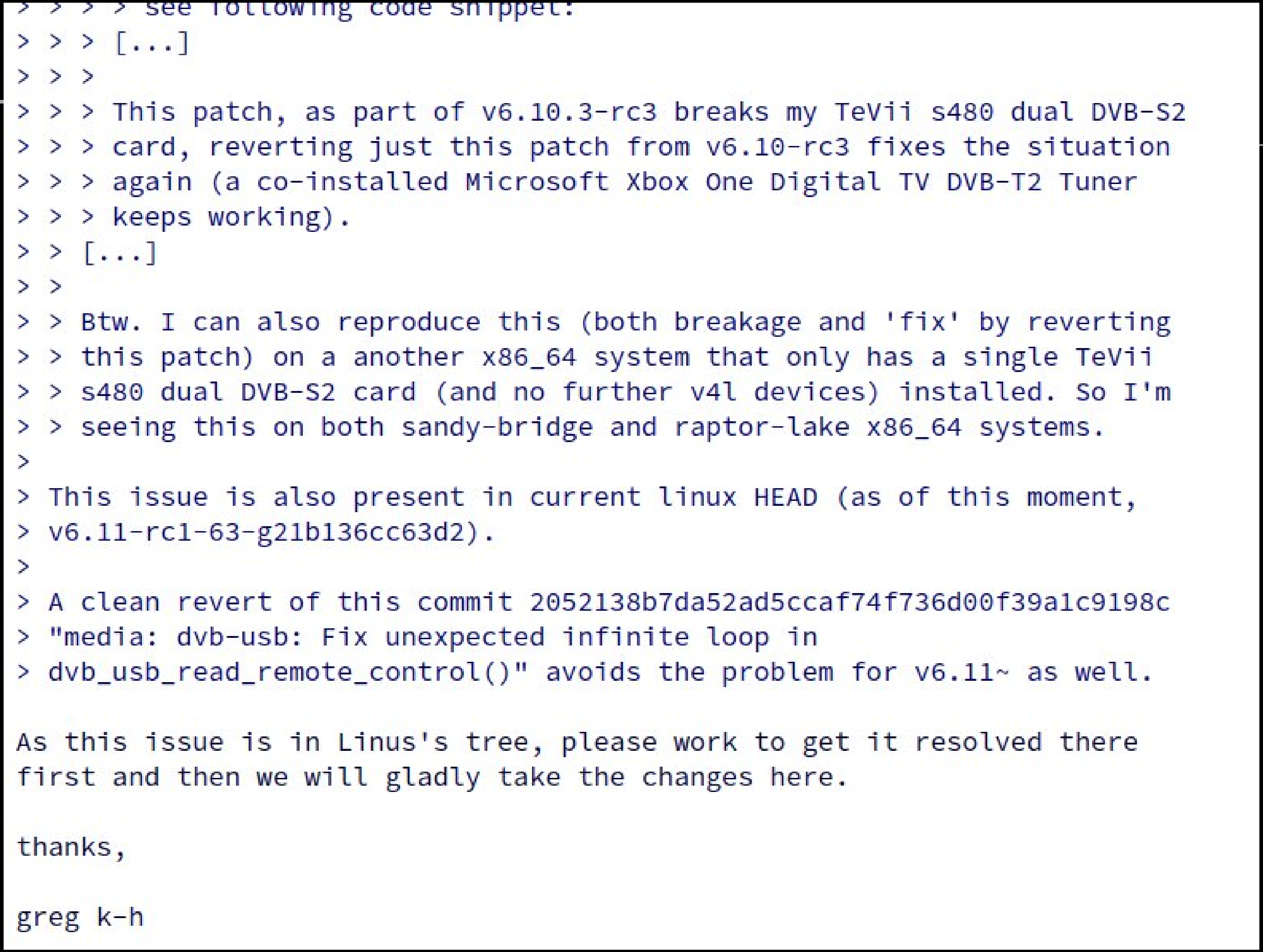
A patch is merged after 6.10-rc1; on Wednesday it is
backported to 6.9.3 and right afterwards someone reports
it causes a regression in mainline and stable a revert is able to fix;
a straight-forward fix is reviewed by Thursday morning;
during Thursday it then is committed to a subsystem tree;
it makes it to -next a day later;
it after five weeks in -next is then send to Linus;
Linus picks it up for 6.10-rc7.
The fix contains a 'Fixes:…' tag, but not 'CC: <stable@…';
it a few days later is backported to 6.9.y.

A patch is merged after 6.10-rc1; on Wednesday it is backported to 6.9.3 and right afterwards someone reports it causes a regression in mainline and stable a revert is able to fix; a straight-forward fix is reviewed by Thursday morning; during Thursday it then is committed to a subsystem tree; it makes it to -next a day later; it after five weeks in -next is then send to Linus; Linus picks it up for 6.10-rc7. The fix contains a 'Fixes:…' tag, but not 'CC: <stable@…'; it a few days later is backported to 6.9.y.

workflow problem: mainline devs do not have to care about stable and might wait till the end of the cycle to fix the problem

A patch is merged after 6.10-rc1; on Wednesday it is backported to 6.9.3 and right afterwards someone reports it causes a regression in mainline and stable a revert is able to fix; a straight-forward fix is reviewed by Thursday morning; during Thursday it then is committed to a subsystem tree; it makes it to -next a day later; it after five weeks in -next is then send to Linus; Linus picks it up for 6.10-rc7. The fix contains a 'Fixes:…' tag, but not 'CC: <stable@…'; it a few days later is backported to 6.9.y.

workflow oddity: stable team normally does not revert a backported change, if it causes the same problem in mainline



```
> > > > see following code snippet:
> > > [...]
> > >
> > > This patch, as part of v6.10.3-rc3 breaks my TeVii s480 dual DVB-S2
> > > card, reverting just this patch from v6.10-rc3 fixes the situation
> > > again (a co-installed Microsoft Xbox One Digital TV DVB-T2 Tuner
> > > keeps working).
> > [...]
> >
> > Btw. I can also reproduce this (both breakage and 'fix' by reverting
> > this patch) on a another x86_64 system that only has a single TeVii
> > s480 dual DVB-S2 card (and no further v4l devices) installed. So I'm
> > seeing this on both sandy-bridge and raptor-lake x86_64 systems.
>
> This issue is also present in current linux HEAD (as of this moment,
> v6.11-rc1-63-g21b136cc63d2).
>
> A clean revert of this commit 2052138b7da52ad5ccaf74f736d00f39a1c9198c
> "media: dvb-usb: Fix unexpected infinite loop in
> dvb_usb_read_remote_control()" avoids the problem for v6.11~ as well.

As this issue is in Linus's tree, please work to get it resolved there
first and then we will gladly take the changes here.

thanks,

greg k-h
```

https://lore.kernel.org/all/2024080325-blaming-lid-5f0d@gregkh/

Thorsten's wishlist

guielines or something like that that describes Linus' expectations for everyone involved

wishlist (1/4):

- developers…
  - …react quickly to regressions reports
  - …provide at least some debugging aid when needed
  - …prioritize resolving mainline regressions over almost all other upstream work
  - …prioritize regressions even more if they recently made it into a release deemed for end users
  - …CCs the regression list when replying [bonus points]
  - …tell regzbot about the report [many bonus points]

wishlist (2/4):

- developers…
  - …try to quickly provide a fix
  - opt for a revert if a fix comes not in sight within few days
  - …add all tags required, recommended, or helpful:

    Reported-by: for all reports
    Link: / or Closes: for all reports
    Tested-by:
    Fixes: for all commits directly or indirectly fixed
    Cc: <stable@vger.kernel.org> [when appropriate]
    [Cc: <regressions@lists.kernel.org>?]

wishlist (3/4):

- reviewers…
  - …try to quickly review fixes
  - …prioritize regression fixes
  - …check or the stuff mentioned earlier

wishlist (4/4):

- maintainers…
  - …commit reviewed fixes for recent regressions quickly
  - …when needed mainline urgent fixes within a day or two or ask Linus
  - …usually mainline all fixes at the end of the week (Fri/Sat/Sun)
  - …sometimes even mainline fixes that not have been in -next
  - …have -fixes and -for-next branches that both are in –next [bonus points]
  - …only queue really dangerous fixes or fixes for non-recent regressions for the next merge window

that's it; questions?

Thorsten Leemhuis

mail: linux@leemhuis.info
GPG Key: 0x72B6E6EF4C583D2D

#fediverse: @kernellogger@fosstodon.org (en),
@knurd42@social.linux.pizza (en)

#EOF