# Linux: Kernel Release Number, Part II

Posted by jeremy on Friday, March 4, 2005 - 07:05

In the continued discussion on release numbering for the Linux kernel [story], Linux creator Linus Torvalds decided against trying to add meaning to the odd/even least significant number. Instead, the new plan is to go from the current 2.6.x numbering to a finer-grained 2.6.x.y. Linus will continue to maintain only the 2.6.x releases, and the -rc releases in between. Others will add trivial patches to create the 2.6.x.y releases. Linus cautions that the task of maintaining a 2.6.x.y tree is not going to be enjoyable:

> "I'll tell you what the problem is: I don't think you'll find anybody to do the parallell 'only trivial patches' tree. They'll go crazy in a couple of weeks. Why? Because it's a _damn_ hard problem. Where do you draw the line? What's an acceptable patch? And if you get it wrong, people will complain _very_ loudly, since by now you've 'promised' them a kernel that is better than the mainline. In other words: there's almost zero glory, there are no interesting problems, and there will absolutely be people who claim that you're a dick-head and worse, probably on a weekly basis."

He went on to add, "*that said, I think in theory it's a great idea. It might even be technically feasible if there was some hard technical criteria for each patch that gets accepted, so that you don't have the burn-out problem.*" His suggested criteria included limiting the patch to being 100 lines or less, and requiring that it fix an oops, a hang, or an exploitable security hole. He also proposed a vetting process in which multiple people were involved in deciding what goes into the tree, any one of which could veto any single patch. And finally, the 2.6.x.y tree would only be maintained until 2.6.(x+1) was released, at which time 2.6.x.y would be completely frozen. During all this time, any fixes in the 2.6.x.y branch would be regularly merged into Linus' mainline tree. The proposal met favorable responses, and Greg KH volunteered to begin maintainership.

http://kerneltrap.org/node/4800

```
From: Linus Torvalds [email blocked]

To: Jeff Garzik [email blocked]

Subject: Re: RFD: Kernel release numbering

Date:       Thu, 3 Mar 2005 08:23:39 -0800 (PST)

On Thu, 3 Mar 2005, Jeff Garzik wrote:
>
> Greg KH wrote:
> > Sure they've been asking for it, but I think they really don't know what
> > it entails. Look at all of the "non-stable" type patches in the -ac and
> > as tree. There's a lot of stuff in there. It's a slippery slope down
> > when trying to say, "I'm only going to accept bug fixes."
>
> We have all these problems precisely because _nobody_ is saying "I'm
> only going to accept bug fixes". We _need_ some amount of release
> engineering. Right now we basically have none.
```

I agree that this is one of the main problems.

But look at how to solve it. The _logical_ solution is to have a third line of defense: we have the -mm trees (wild and wacky patches), and we have my tree (hopefully not wacky any more), and it would be good to have a third level tree (which I'm just not interested in, because that one doesn't do any development any more) which only takes the "so totally not wild that it's really boring" patches.

In fact, if somebody maintained that kind of tree, especially in BK, it would be trivial for me to just pull from it every once in a while (like ever _day_ if necessary). But for that to work, then that tree would have to be about so _obviously_ not wild patches that it's a no-brainer.

So what's the problem with this approach? It would seem to make everybody happy: it would reduce my load, it would give people the alternate "2.6.x base kernel plus fixes only" parallell track, and it would _not_ have the testability issue (because I think a lot of people would be happy to test that tree, and if it was always based on the last 2.6.x release, there would be no issues.

Anybody?

I'll tell you what the problem is: I don't think you'll find anybody to do the parallell "only trivial patches" tree. They'll go crazy in a couple of weeks. Why? Because it's a _damn_ hard problem. Where do you draw the line? What's an acceptable patch? And if you get it wrong, people will complain _very_ loudly, since by now you've "promised" them a kernel that is better than the mainline. In other words: there's almost zero glory, there are no interesting problems, and there will absolutely be people who claim that you're a dick-head and worse, probably on a weekly basis.

That said, I think in theory it's a great idea. It might even be technically feasible if there was some hard technical criteria for each patch that gets accepted, so that you don't have the burn-out problem.

So let's loook at how we could set that up. We need:

- a sucker who wants to do this, or a company that pays for somebody good to do this (and remember: "good" here doesn't necessarily have to mean technical genius, it's about taking abuse and being stable). The whole setup should be such that there can never be any question about the patches for _other_ reasons (to avoid the sucker becoming a target for abuse), so this person really to some degree would be fairly mechanical.

Don't make it automated, though. That just gets us down the path of flaming about the scripts and automation. And I'm not claiming that we should aim for somebody _stupid_, I'm just claiming that it takes a certain kind of person to do something that is not all that glamorous, and that puts you in the spot.

We don't ever want to have that spark of "wouldn't this be cool" in this project.

- some very _technical_ and objective rules on patches. And they should limit the patches severely, so that people can never blame the sucker who does the job. For example, I would suggest that "size" be one hard technical rule. If the patch is more than 100 lines (with context) in size, it's not trivial any more. Really. Two big screenfuls (or four, for people who still use the ISO-ANSI standard 80x24 vt100)

Also, I'd suggest that a _hard_ rule (ie nobody can override it) would also be that the problem causes an oops, a hang, or a real security

problem that somebody can come up with an exploit for (ie no "there could be a two-instruction race" crap. Only "there is a race, and here's how you exploit it"). The exploit wouldn't need to be full code that gets root, but an explanation of it, at least.

- a vetting process. You'd have ten people, and five of them would have to sign off on the patch, and even a single veto would shoot it down.

Again, this is really to protect the sucker, and make it possible to work: I don't think this can work with a creative person (everybody else calls me "flaky", and I much prefer that "creative" word, it sounds so much better), which I personally believe means that we don't _want_ people like Alan, Andrea, Andrew etc etc that have historically maintained their own trees that sometimes have tried to do something like this.

- Finally: this tree never has any history past the "last release". When a new kernel comes, the tree is frozen, and never to be touched again.

If somebody _else_ wants to base things off this special "sucker tree", and make a fourth level tree that is based on the _previous_ stable tree, that's fine, but that's a separate process. He would be totally free to do so, but the rule is that this particular maintenance program _never_ gets stuck on an old kernel, like the vendor trees always are.

This is not a long-range tree, it would _purely_ be about one thing and one thing only: the last stable kernel. The people involved (sucker and vetters all) would never have to remember two different trees, or care about problems that aren't in the top-of-tree. Keep ti simple, and keep the rules clear.

Does this mean that some patches would never go into this tree? Yes. It would mean that patches that some people might feel very _strongly_ are good patches would never ever show up in this tree, but on the other hand, I can see this tree being useful regardless, and I think the lack of flexibility in this case is actually the whole _point_ of the tree. The lack of flexibility is the very thing that makes this be the kind of base that anybody else can then hang their own patches on top of. There should never be a situation where "I'd like that tree, but I think xxxx was done wrong".

Might something like this make people happier? (I wrote "happy" rather than "happier" at first, but let's face it, people are better at whining than they are at being happy ;)

Linus

From: Greg KH [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 08:43:53 -0800

On Thu, Mar 03, 2005 at 08:23:39AM -0800, Linus Torvalds wrote:
>
> So what's the problem with this approach? It would seem to make everybody
> happy: it would reduce my load, it would give people the alternate "2.6.x
> base kernel plus fixes only" parallell track, and it would _not_ have the
> testability issue (because I think a lot of people would be happy to test
> that tree, and if it was always based on the last 2.6.x release, there
> would be no issues.
>
> Anybody?

Well, I'm one person who has said that this would be a very tough problem to solve. And hey, I like tough problems, so I'll volunteer to start this. If I burn out, I'll take the responsibility of finding

someone else to take it over.

I really like the rules you've outlined, that makes it almost possible

to achieve sanity.

thanks,

greg k-h

From: Chris Friesen [email blocked]

Subject: Re: RFD: Kernel release numbering

Date: Thu, 03 Mar 2005 10:55:13 -0600

Linus Torvalds wrote:

> I'll tell you what the problem is: I don't think you'll find anybody to do
> the parallell "only trivial patches" tree.

Isn't this what -ac and -as effectively already are?

Chris

From: Greg KH [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 08:59:40 -0800

On Thu, Mar 03, 2005 at 10:55:13AM -0600, Chris Friesen wrote:
> Linus Torvalds wrote:
>
> >I'll tell you what the problem is: I don't think you'll find anybody to do
> >the parallell "only trivial patches" tree.
>
> Isn't this what -ac and -as effectively already are?

Based on the patches in those trees, no :)

thanks,

greg k-h

From: Alan Cox [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 03 Mar 2005 23:51:42 +0000

On Iau, 2005-03-03 at 16:59, Greg KH wrote:
> On Thu, Mar 03, 2005 at 10:55:13AM -0600, Chris Friesen wrote:
> > Linus Torvalds wrote:
> >
> > >I'll tell you what the problem is: I don't think you'll find anybody to do
> > >the parallell "only trivial patches" tree.
> >
> > Isn't this what -ac and -as effectively already are?
>
> Based on the patches in those trees, no :)

I've not found a much smaller set that isn't rootable, trivially DoSable
with published tools or leaves users with non-working hardware that got
pulled by Linus having a random pissy fit about pwc etc.

-ac is essentially base security fixes + working IDE locking + pwc +
fixes for the bugs everyone hit that needed fixing urgently. I consider
working locking on my storage essential because I like my data to still
be there.

-as is similar although it makes different choices about what matters.

Given 3 or 4 people it ought to be possible to make a much much tighter
patch set for this purpose.

From: Linus Torvalds [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 09:50:29 -0800 (PST)

On Thu, 3 Mar 2005, Chris Friesen wrote:
>
> Linus Torvalds wrote:
>
> > I'll tell you what the problem is: I don't think you'll find anybody to do
> > the parallell "only trivial patches" tree.
>
> Isn't this what -ac and -as effectively already are?

No. They both end up doing a lot of much fancier stuff. There are patches
in there that I may not be comfortable with, because they end up doing
things like totally re-doing the locking for some subsystem.

Yes, they end up re-doing _broken_ locking, but the point is that they are
not obvious. They are just "more careful" versions of the -mm tree.

Linus


From: Chris Wright [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 08:55:33 -0800

* Linus Torvalds (torvalds@osdl.org) wrote:
> On Thu, 3 Mar 2005, Jeff Garzik wrote:
> In fact, if somebody maintained that kind of tree, especially in BK, it
> would be trivial for me to just pull from it every once in a while (like
> ever _day_ if necessary). But for that to work, then that tree would have
> to be about so _obviously_ not wild patches that it's a no-brainer.
>
> So what's the problem with this approach? It would seem to make everybody
> happy: it would reduce my load, it would give people the alternate "2.6.x
> base kernel plus fixes only" parallell track, and it would _not_ have the
> testability issue (because I think a lot of people would be happy to test
> that tree, and if it was always based on the last 2.6.x release, there
> would be no issues.
>
> Anybody?

Andres Salomon (-as patches) and I have been talking about that at least
regarding security fixes. It's worth trying in a more complete and
formalized way. Guess I can be branded a sucker ;-)

> I'll tell you what the problem is: I don't think you'll find anybody to do
> the parallell "only trivial patches" tree. They'll go crazy in a couple of
> weeks. Why? Because it's a _damn_ hard problem. Where do you draw the
> line? What's an acceptable patch? And if you get it wrong, people will
> complain _very_ loudly, since by now you've "promised" them a kernel that
> is better than the mainline. In other words: there's almost zero glory,
> there are no interesting problems, and there will absolutely be people who

> claim that you're a dick-head and worse, probably on a weekly basis.
>
> That said, I think in theory it's a great idea. It might even be
> technically feasible if there was some hard technical criteria for each
> patch that gets accepted, so that you don't have the burn-out problem.
>
> So let's loook at how we could set that up. We need:
>
> - a sucker who wants to do this, or a company that pays for somebody good
> to do this (and remember: "good" here doesn't necessarily have to mean
> technical genius, it's about taking abuse and being stable). The whole
> setup should be such that there can never be any question about the
> patches for _other_ reasons (to avoid the sucker becoming a target for
> abuse), so this person really to some degree would be fairly
> mechanical.
>
> Don't make it automated, though. That just gets us down the path of
> flaming about the scripts and automation. And I'm not claiming that we
> should aim for somebody _stupid_, I'm just claiming that it takes a
> certain kind of person to do something that is not all that glamorous,
> and that puts you in the spot.
>
> We don't ever want to have that spark of "wouldn't this be cool" in
> this project.
>
> - some very _technical_ and objective rules on patches. And they should
> limit the patches severely, so that people can never blame the sucker
> who does the job. For example, I would suggest that "size" be one hard
> technical rule. If the patch is more than 100 lines (with context) in
> size, it's not trivial any more. Really. Two big screenfuls (or four,
> for people who still use the ISO-ANSI standard 80x24 vt100)
>
> Also, I'd suggest that a _hard_ rule (ie nobody can override it) would
> also be that the problem causes an oops, a hang, or a real security
> problem that somebody can come up with an exploit for (ie no "there
> could be a two-instruction race" crap. Only "there is a race, and
> here's how you exploit it"). The exploit wouldn't need to be full code
> that gets root, but an explanation of it, at least.
>
> - a vetting process. You'd have ten people, and five of them would have
> to sign off on the patch, and even a single veto would shoot it down.
>
> Again, this is really to protect the sucker, and make it possible to
> work: I don't think this can work with a creative person (everybody
> else calls me "flaky", and I much prefer that "creative" word, it sounds
> so much better), which I personally believe means that we don't _want_
> people like Alan, Andrea, Andrew etc etc that have historically maintained
> their own trees that sometimes have tried to do something like this.
>
> - Finally: this tree never has any history past the "last release". When
> a new kernel comes, the tree is frozen, and never to be touched again.

I like this definition. The only remaining question is what determines
a 2.6.x.y release? One patch? Sure if it's critical enough.

> If somebody _else_ wants to base things off this special "sucker tree",
> and make a fourth level tree that is based on the _previous_ stable
> tree, that's fine, but that's a separate process. He would be totally
> free to do so, but the rule is that this particular maintenance program
> _never_ gets stuck on an old kernel, like the vendor trees always are.
>
> This is not a long-range tree, it would _purely_ be about one thing and
> one thing only: the last stable kernel. The people involved (sucker and
> vetters all) would never have to remember two different trees, or care

> about problems that aren't in the top-of-tree. Keep ti simple, and keep
> the rules clear.
>
> Does this mean that some patches would never go into this tree? Yes. It
> would mean that patches that some people might feel very _strongly_ are
> good patches would never ever show up in this tree, but on the other hand,
> I can see this tree being useful regardless, and I think the lack of
> flexibility in this case is actually the whole _point_ of the tree. The
> lack of flexibility is the very thing that makes this be the kind of base
> that anybody else can then hang their own patches on top of. There should
> never be a situation where "I'd like that tree, but I think xxxx was done
> wrong".
>
> Might something like this make people happier? (I wrote "happy" rather
> than "happier" at first, but let's face it, people are better at whining
> than they are at being happy ;)

Heh, maybe people are happiest when they are whining ;-)

thanks,
-chris
--
Linux Security Modules http://lsm.immunix.org http://lsm.bkbits.net

From: Jens Axboe [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 18:03:36 +0100

On Thu, Mar 03 2005, Chris Wright wrote:
> > - Finally: this tree never has any history past the "last release". When
> > a new kernel comes, the tree is frozen, and never to be touched again.
>
> I like this definition. The only remaining question is what determines
> a 2.6.x.y release? One patch? Sure if it's critical enough.

Why should there be one? One of the things I like about this concept is
that it's just a moving tree. There could be daily snapshots like the
-bkX "releases" of Linus's tree, if there are changes from the day
before. It means (hopefully) that no one will "wait for x.y.z.2 because
that is really stable".

--
Jens Axboe

From: Greg KH [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 09:07:33 -0800

On Thu, Mar 03, 2005 at 06:03:36PM +0100, Jens Axboe wrote:
> On Thu, Mar 03 2005, Chris Wright wrote:
> > > - Finally: this tree never has any history past the "last release". When
> > > a new kernel comes, the tree is frozen, and never to be touched again.
> >
> > I like this definition. The only remaining question is what determines
> > a 2.6.x.y release? One patch? Sure if it's critical enough.
>
> Why should there be one? One of the things I like about this concept is
> that it's just a moving tree. There could be daily snapshots like the
> -bkX "releases" of Linus's tree, if there are changes from the day
> before. It means (hopefully) that no one will "wait for x.y.z.2 because
> that is really stable".

So that means we do the release often, _and_ provide -bkX snapshots
daily if we happen to take a few days between releases and patches are

pending for some reason.

thanks,

greg k-h

From: Linus Torvalds [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 09:53:40 -0800 (PST)

On Thu, 3 Mar 2005, Jens Axboe wrote:
>
> Why should there be one? One of the things I like about this concept is
> that it's just a moving tree. There could be daily snapshots like the
> -bkX "releases" of Linus's tree, if there are changes from the day
> before. It means (hopefully) that no one will "wait for x.y.z.2 because
> that is really stable".

Exactly. Th ewhole point of this tree is that there shouldn't be anything
questionable in it. All the patches are independent, and they are all
trivial and small.

Which is not to say there couldn't be regressions even from trivial and
small patches, and yes, there will be an outcry when there is, but we're
talking minimizing the risk, not making it impossible.

Linus

From: "Sean" [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 14:35:29 -0500 (EST)

On Thu, March 3, 2005 12:53 pm, Linus Torvalds said:
> On Thu, 3 Mar 2005, Jens Axboe wrote:
>>
>> Why should there be one? One of the things I like about this concept is
>> that it's just a moving tree. There could be daily snapshots like the
>> -bkX "releases" of Linus's tree, if there are changes from the day
>> before. It means (hopefully) that no one will "wait for x.y.z.2 because
>> that is really stable".
>
> Exactly. Th ewhole point of this tree is that there shouldn't be anything
> questionable in it. All the patches are independent, and they are all
> trivial and small.
>
> Which is not to say there couldn't be regressions even from trivial and
> small patches, and yes, there will be an outcry when there is, but we're
> talking minimizing the risk, not making it impossible.
>

Wait a second though, this tree will be branched from the development
mainline. So it will contain many patches that entered with less
testing. What will be the policy for dealing with regressions relative
to the previous $sucker release caused by huge patches that entered via
the development tree? Is reverting them prohibited because of the patch
size?

Sean

From: Linus Torvalds [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 12:41:02 -0800 (PST)

On Thu, 3 Mar 2005, Sean wrote:

>
> Wait a second though, this tree will be branched from the development
> mainline. So it will contain many patches that entered with less
> testing.

Well, since I'd pull basically all the time, all the patches _do_ get
testing the the -rc kernels.

But the rules would be that small patches can also be verified
independently.

And realize that the 2.6.x tree doesn't "change" any way. It doesn't get
more unruly just because we have a side tree that is being anal about
things. It's still as stable as it ever is..

Linus


From: Adrian Bunk [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 18:08:08 +0100

On Thu, Mar 03, 2005 at 08:23:39AM -0800, Linus Torvalds wrote:
>...
> But look at how to solve it. The _logical_ solution is to have a third
> line of defense: we have the -mm trees (wild and wacky patches), and we
> have my tree (hopefully not wacky any more), and it would be good to have
> a third level tree (which I'm just not interested in, because that one
> doesn't do any development any more) which only takes the "so totally not
> wild that it's really boring" patches.
>
> In fact, if somebody maintained that kind of tree, especially in BK, it
> would be trivial for me to just pull from it every once in a while (like
> ever _day_ if necessary). But for that to work, then that tree would have
> to be about so _obviously_ not wild patches that it's a no-brainer.
>
> So what's the problem with this approach? It would seem to make everybody
> happy: it would reduce my load, it would give people the alternate "2.6.x
> base kernel plus fixes only" parallell track, and it would _not_ have the
> testability issue (because I think a lot of people would be happy to test
> that tree, and if it was always based on the last 2.6.x release, there
> would be no issues.
>...
> - some very _technical_ and objective rules on patches. And they should
> limit the patches severely, so that people can never blame the sucker
> who does the job. For example, I would suggest that "size" be one hard
> technical rule. If the patch is more than 100 lines (with context) in
> size, it's not trivial any more. Really. Two big screenfuls (or four,
> for people who still use the ISO-ANSI standard 80x24 vt100)
>...

This only attacks part of the problem.

Most regressions that annoy users aren't in some core system, they are
in drivers.

What if $big_driver_update in 2.6.12 fixed a serious bug for some people
but broke the driver for many other people, and both reverting this
patch and fixing the driver for the people it's broken for exceeds such
size limits?

> Linus

cu

Adrian

--

"Is there not promise of rain?" Ling Tan asked suddenly out
of the darkness. There had been need of rain for many days.
"Only a promise," Lao Er said.
Pearl S. Buck - Dragon Seed

From: Greg KH [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 09:12:26 -0800

On Thu, Mar 03, 2005 at 06:08:08PM +0100, Adrian Bunk wrote:
> On Thu, Mar 03, 2005 at 08:23:39AM -0800, Linus Torvalds wrote:
> >...
> > But look at how to solve it. The _logical_ solution is to have a third
> > line of defense: we have the -mm trees (wild and wacky patches), and we
> > have my tree (hopefully not wacky any more), and it would be good to have
> > a third level tree (which I'm just not interested in, because that one
> > doesn't do any development any more) which only takes the "so totally not
> > wild that it's really boring" patches.
> >
> > In fact, if somebody maintained that kind of tree, especially in BK, it
> > would be trivial for me to just pull from it every once in a while (like
> > ever _day_ if necessary). But for that to work, then that tree would have
> > to be about so _obviously_ not wild patches that it's a no-brainer.
> >
> > So what's the problem with this approach? It would seem to make everybody
> > happy: it would reduce my load, it would give people the alternate "2.6.x
> > base kernel plus fixes only" parallell track, and it would _not_ have the
> > testability issue (because I think a lot of people would be happy to test
> > that tree, and if it was always based on the last 2.6.x release, there
> > would be no issues.
> >...
> > - some very _technical_ and objective rules on patches. And they should
> > limit the patches severely, so that people can never blame the sucker
> > who does the job. For example, I would suggest that "size" be one hard
> > technical rule. If the patch is more than 100 lines (with context) in
> > size, it's not trivial any more. Really. Two big screenfuls (or four,
> > for people who still use the ISO-ANSI standard 80x24 vt100)
> >...
>
> This only attacks part of the problem.
>
> Most regressions that annoy users aren't in some core system, they are
> in drivers.
>
> What if $big_driver_update in 2.6.12 fixed a serious bug for some people
> but broke the driver for many other people, and both reverting this
> patch and fixing the driver for the people it's broken for exceeds such
> size limits?

Then either the patch author splits out the bug if they want to, or we
punt and say "wait for 2.6.12". Distros can then decide if they want to
take the whole $big_patch in their releases, if they are near a release
cycle.

I feel that imposing such limits is the only way this can be done and
remain sane.

thanks,

greg k-h

From: Linus Torvalds [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 10:02:49 -0800 (PST)

On Thu, 3 Mar 2005, Greg KH wrote:
>
> Then either the patch author splits out the bug if they want to, or we
> punt and say "wait for 2.6.12". Distros can then decide if they want to
> take the whole $big_patch in their releases, if they are near a release
> cycle.

Yes. Exactly.

We're not aiming for "perfect". Just _trying_ to be perfect is what would kill the whole scheme in the first place. We'd be aiming for "known rules".

Whether people _agree_ with those rules is then actually not a huge issue. There will _always_ be things that people don't agree with. Aiming for consistency is worthwhile in itself.

(Of course, the rules _do_ matter in the sense that there has to be some point to the consistency. You can have a consistent rule that "the ChangeLog entries must rhyme", and I think it's a great rule, and I encourage anybody who wants to to set up such a "rhyming kernel tree", but that doesn't mean that it makes a lot of difference to people ;).

So havign strict rules that allow _one_ kind of consistency that people agree is good is a fine idea.

And Adrian, you can always have a different tree that has another set of rules - and if you use BK you can merge the two and have the "combination of the rules" tree. The reason I would _stronly_ urge very tight rules is that if they aren't tight, it ends up having all the problems we've always seen in other trees.

For example, if the "tight rules tree" allowed reverting an otheriwse good patch because it had a bug (instead of trying to fix the bug), then I would never be able to pull that tree into mine. It would take development _backwards_, and thus it might be sensible for a vendor, but it would automatically mean that it's not a good base for the next kernel version.

And if I can't just say "ok, I'll always take the 'tight rules' tree", then we'd get into the forward-and-backward porting hell again, which would make the whole tree totally pointless. See my point?

Linus

From: Thomas Gleixner [email blocked]

Subject: Re: RFD: Kernel release numbering

Date: Thu, 03 Mar 2005 20:15:35 +0100

On Thu, 2005-03-03 at 18:08 +0100, Adrian Bunk wrote:

> This only attacks part of the problem.

It still does not solve the problem of "untested" releases. Users will still ignore the linus-tree-rcX kernels.

So we move the real -rcX phase after the so called stable release.

Doing -rcX from the "sucker" tree up to a stable release makes much more sense and would have more testers and get back lost confidence.

tglx

From: Linus Torvalds [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 11:37:35 -0800 (PST)

On Thu, 3 Mar 2005, Thomas Gleixner wrote:
>
> It still does not solve the problem of "untested" releases. Users will
> still ignore the linus-tree-rcX kernels.

.. and maybe that problem is unsolvable. People certainly argued vehemently that anything we do to try to make test releases (renaming etc) won't help.

So what do you do if you find an unsolvable problem? You don't solve it: you make sure it's not a show-stopper.

So part of the idea of having the "other tree" is that it ends up solving the "hmm, we missed that detail" problem. And by _not_ giving it release numbers or any schedule at all, people can't _wait_ for it.

Sneaky. That's my middle name.

Linus

From: Andrew Morton [email blocked]

Subject: Re: RFD: Kernel release numbering

Date: Thu, 3 Mar 2005 15:17:52 -0800

Greg KH [email blocked] wrote:
>
> > It's perfectly workable from a BK standpoint to do
> >
> > -> linux-2.6 commit
> > -> cpcset into linux-2.6.X.Y [see Documentation/BK-usage/cpcset]
> > -> pull from linux-2.6.X.Y into linux-2.6 [dups cset, but no
> > real code change]
>
> That's fine with me to do. As long as someone points out to $sucker
> that such a patch should go into 2.6.x.y.

That's the only way it _can_ work. The maintainer of 2.6.x.y shouldn't be put in a position of having to locate the patches which he needs.

Like it or not, Vojtech is still the maintainer of the input system in 2.6.x.y and he should be the primary guy who keeps an eye out for patches which needs to be applied there.

If we go overboard here, every maintainer will end up maintaining another tree of "stuff which needs to be backported to 2.6.x.y", which is probably more work than we want to do.

As long as we can keep it down to "small and really critical things" then it should work OK.

Ideally, the 2.6.x.y maintainer wouldn't need any particular kernel
development skills - it's just patchmonkeying the things which maintainers
send him.

From: Alan Cox [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Fri, 04 Mar 2005 00:01:52 +0000

On Iau, 2005-03-03 at 23:17, Andrew Morton wrote:
> Ideally, the 2.6.x.y maintainer wouldn't need any particular kernel
> development skills - it's just patchmonkeying the things which maintainers
> send him.

I would disagree, and I suspect anyone else who has maintained a distro
stable kernel would likewise. It needs one or more people who know who
to ask about stuff, are careful, have a good grounding in bug spotting,
races, common mistakes and know roughly how all the kernel works.
Maintainers aren't very good at it in general and they don't see
overlaps between areas very well.

Realistically you have to do stuff like read each Linus checkin and
classify it.

Al Viro is probably the perfect 2.6.x.y maintainer but I doubt he'd want
to do it.

From: Andrew Morton [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 18:28:20 -0800

Alan Cox [email blocked] wrote:
>
> On Iau, 2005-03-03 at 23:17, Andrew Morton wrote:
> > Ideally, the 2.6.x.y maintainer wouldn't need any particular kernel
> > development skills - it's just patchmonkeying the things which maintainers
> > send him.
>
> I would disagree, and I suspect anyone else who has maintained a distro
> stable kernel would likewise. It needs one or more people who know who
> to ask about stuff, are careful, have a good grounding in bug spotting,
> races, common mistakes and know roughly how all the kernel works.
> Maintainers aren't very good at it in general and they don't see
> overlaps between areas very well.
>

That is all inappropriate activity for a 2.6.x.y tree as it is being
proposed.

Am I right? All we're proposing here is a tree which has small fixups for
reasonably serious problems. Almost without exception it would consist of
backports.

From: Alan Cox [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Fri, 04 Mar 2005 10:56:46 +0000

On Gwe, 2005-03-04 at 02:28, Andrew Morton wrote:
> > I would disagree, and I suspect anyone else who has maintained a distro
> > stable kernel would likewise. It needs one or more people who know who
> > to ask about stuff, are careful, have a good grounding in bug spotting,
> > races, common mistakes and know roughly how all the kernel works.
> > Maintainers aren't very good at it in general and they don't see
> > overlaps between areas very well.

> >
> That is all inappropriate activity for a 2.6.x.y tree as it is being
> proposed.
>
> Am I right? All we're proposing here is a tree which has small fixups for
> reasonably serious problems. Almost without exception it would consist of
> backports.

Almost without exception maintainers will forget the backport (there are
some notable exceptions). Almost without exception maintainers will not
be aware that their backport fix clashes with another fix because that
isn't their concern.

Linus will try and sneak stuff in that is security but not mentioned
which has to be dug out (because the bad guys read the patches too).

And finally Linus throws the occasional gem into the backporting mix
because he will (rightly) do the long term fix that rearranges a lot of
code when the .x.y patch needs to be the ugly band aid.

So for example Linus will happily changed remap_vm_area to fix a
security bug by changing the API entirely and making it do some other
things. Or in the case of the exec bug he did a fix that defaulted any
missed fixes to unsafe. Fine for upstream where the goal is cleanness,
bad for .x.y because the arch people hadn't caught up and did have
remaining holes.

You also have to review the dependancy tree for a backport and what was
tested - so I skipped the NFS df fix as one example as it had never been
tested standalone only on a pile of other NFS fixes.

Alan

From: Andrew Morton [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Fri, 4 Mar 2005 03:28:20 -0800

Alan Cox [email blocked] wrote:
>
> Almost without exception maintainers will forget the backport (there are
> some notable exceptions). Almost without exception maintainers will not
> be aware that their backport fix clashes with another fix because that
> isn't their concern.
>
> Linus will try and sneak stuff in that is security but not mentioned
> which has to be dug out (because the bad guys read the patches too).
>
> And finally Linus throws the occasional gem into the backporting mix
> because he will (rightly) do the long term fix that rearranges a lot of
> code when the .x.y patch needs to be the ugly band aid.
>
> So for example Linus will happily changed remap_vm_area to fix a
> security bug by changing the API entirely and making it do some other
> things. Or in the case of the exec bug he did a fix that defaulted any
> missed fixes to unsafe. Fine for upstream where the goal is cleanness,
> bad for .x.y because the arch people hadn't caught up and did have
> remaining holes.
>
> You also have to review the dependancy tree for a backport and what was
> tested - so I skipped the NFS df fix as one example as it had never been
> tested standalone only on a pile of other NFS fixes.

I think you're assuming that 2.6.x.y will have larger scope than is intended.

From: Alan Cox [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Fri, 04 Mar 2005 12:51:29 +0000

On Gwe, 2005-03-04 at 11:28, Andrew Morton wrote:
> I think you're assuming that 2.6.x.y will have larger scope than is intended.

The examples I gave for remap_vm_area and exec are both from real world "gosh look I am root isn't that fun" type security holes. If that is outside the scope of 2.6.x.y then you need to go back to the drawing board.

From: Andrew Morton [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Fri, 4 Mar 2005 05:05:26 -0800

Alan Cox [email blocked] wrote:
>
> On Gwe, 2005-03-04 at 11:28, Andrew Morton wrote:
> > I think you're assuming that 2.6.x.y will have larger scope than is intended.
>
> The examples I gave for remap_vm_area and exec are both from real world
> "gosh look I am root isn't that fun" type security holes. If that is
> outside the scope of 2.6.x.y then you need to go back to the drawing
> board.

Well *obviously* things like that are in scope.

It's hardly likely that "maintainers will forget the backport" in that case, is it?

From: Linus Torvalds [email blocked]

Subject: Re: RFD: Kernel release numbering

Date: Thu, 3 Mar 2005 10:27:12 -0800 (PST)

On Thu, 3 Mar 2005, Jeff Garzik wrote:
>
> The only problem I see with this -- and its a minor problem -- is that
> some patches that belong in the 2.6.X.Y tree go straight to you/Andrew,
> rather than to $sucker.

Yes. I think people will have to be taught, and get used to the new world order, and that could take a long time. And don't get me wrong, I include myself in those people, ie it's not just that everybody else needs to learn to Cc: the new group (I assume it's best to have a mailing alias, to allow the thing to have multiple people involved even before it gets to the vetting stage, and then have a _separate_ mail alias for the "vettign group" people).

Think of how the -mm tree has evolved - with me and Andrew learning how the other side acts and works. This would be the same thing, except hopefully on a smaller scale (ie the _volume_ of patches had better be an order of magnited smaller not just in size but in number too). It wasn't just "let's set up Andrew". It was a learning experience.

And yes, we'll probably get duplicated changes, _especially_ early on. But at least nobody seems to hate this idea, so I think we should drop the original even/odd suggestion for now, and see if this would make more sense..

Linus

From: "Hua Zhong" [email blocked]
Subject: RE: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 10:42:00 -0800

> In fact, if somebody maintained that kind of tree, especially
> in BK, it would be trivial for me to just pull from it every once in a
> while (like ever _day_ if necessary). But for that to work, then that
> tree would have to be about so _obviously_ not wild patches that
> it's a no-brainer.

Alan Cox once said he would like to do it:

> On Mer, 2004-10-27 at 19:37, Hua Zhong wrote:
> When I said "nobody", I really meant "top kernel developers". I have
not
> seen anyone step up and say "I'll volunteer to maintain a 2.6 stable
> release" hence the comment.

I'll do it if Linus wants

Do you consider having a real stable release maintainer again?

If you want someone to do the job, give him a title. It's a thankless and boring job, and you can't make it worse by just hiding him somewhere.

How a "stable release maintainer" works for the current model is up to you. One thought is that he picks up 2.6.x release as a start point and takes patches to make it stable, then releases it _himself_, not by Linus. Because the real work is done by him and you need to give him the authority (just like other Linux 2.x maintainers who release official kernels). But of course you still pull from his tree to make sure the bug fixes are also committed to mainline.

Hua

From: Linus Torvalds [email blocked]
Subject: RE: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 11:11:05 -0800 (PST)

On Thu, 3 Mar 2005, Hua Zhong wrote:
>
> Do you consider having a real stable release maintainer again?

No, this really is a different thing.

This is not a "truly separate" parallell track, exactly because it would not actually get a life of its own. For it to make sense, it would not do any big changes, ie it would be _limited_ in a way that a real stable release would not. Also, since it would leave the old kernel behind when a new stable release comes along, it would not have any real independence in time either.

Now, I think this "sucker tree" I'm talking about would be a great basis for somebody else then taking it _further_ (ie vendor stable trees), but it really is a fairly small step.

> If you want someone to do the job, give him a title. It's a thankless and
> boring job, and you can't make it worse by just hiding him somewhere.

Actually, that was something I'd _avoid_ - make it non-glorious on

purpose. In the kind of tree I envision, the _last_ thing we'd want is the maintainer looking at a big picture and feeling important. I'd be happiest if he was almost totally anonymous, because I think it's likely a boring job, but it's a boring job that _many_ people could do (ie to avoid burnign people out, make it be a stint of a couple of months, not a

"crowning life work", and then you could probably have half a dozen people who are perfectly willing to take it on every once in a while.

Ie I'd organize it like some of the "checkin committees" work for other projects that have nowhere _near_ as much work going on as Linux has. That seems to work well for small projects - and we can try to keep this "small" exactly by having the strict rules in place that would mean that 99% of all patches wouldn't even be a consideration.

In other words, I'm really talking about something different from what you seem to envision. I think we should call the tree the "sucker tree", and if somebody wants to make a logo for it, make it be a penguin with a jokers' hat: exactly to remind people that it's not about the glory.

(Maybe that's going overboard a bit ;)

Linus

From: Alan Cox [email blocked]

Subject: Re: RFD: Kernel release numbering

Date: Thu, 03 Mar 2005 22:15:46 +0000

On Mer, 2005-03-02 at 22:21, Linus Torvalds wrote:
> - 2.6.<odd>: still a stable kernel, but accept bigger changes leading up
> to it (timeframe: a month or two).
> - 2.<odd>.x: aim for big changes that may destabilize the kernel for
> several releases (timeframe: a year or two)
> - <odd>.x.x: Linus went crazy, broke absolutely _everything_, and rewrote
> the kernel to be a microkernel using a special message-passing version
> of Visual Basic. (timeframe: "we expect that he will be released from
> the mental institution in a decade or two").

We still need 2.6.x.y updates on a more official footing and with more than one person as the "2.6.x.y" maintainer. I think that is actually more important.

The 2.6.<odd> thing is essentially irrelevant. You are just relabelling pre
and rc to even and odd. It won't fool anyone into testing it more....

From: Jeff Garzik [email blocked]
Subject: Re: RFD: Kernel release numbering
Date: Thu, 3 Mar 2005 17:32:03 -0500

On Thu, Mar 03, 2005 at 10:15:46PM +0000, Alan Cox wrote:
> We still need 2.6.x.y updates on a more official footing and with more
> than one person as the "2.6.x.y" maintainer. I think that is actually
> more important.

That appears to be the consensus conclusion we've arrived at.

Jeff