

A better way to review
mutable history

Who am I?

Jordi Gutiérrez Hermoso

Who am I?

Jordi Gutiérrez Hermoso



Grist

Email: jordi@getgrist.com

Fediverse: [@jordigh@mathstodon.xyz](https://mathstodon.xyz/@jordigh)



Who am I?

Jordi Gutiérrez Hermoso



Grist

Email: jordi@getgrist.com

Fediverse: [@jordigh@mathstodon.xyz](https://mathstodon.xyz/@jordigh)



Who am I?

Jordi Gutiérrez Hermoso

- NES enthusiast (let's talk 6502)
- Mercurial contributor (still using it!)
- Grist systems developer



Grist

Email: jordi@getgrist.com

Fediverse: [@jordigh@mathstodon.xyz](https://mathstodon.xyz/@jordigh)



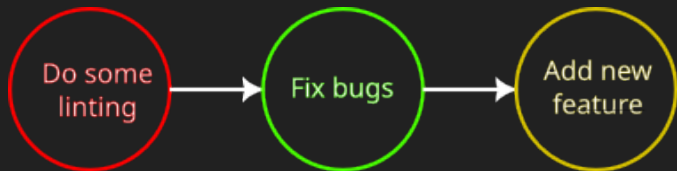
Code review today

Most common method popularised by Github

Code review today

Most common method popularised by Github

- Pull request — the unit of review



Code review today

Most common method popularised by Github

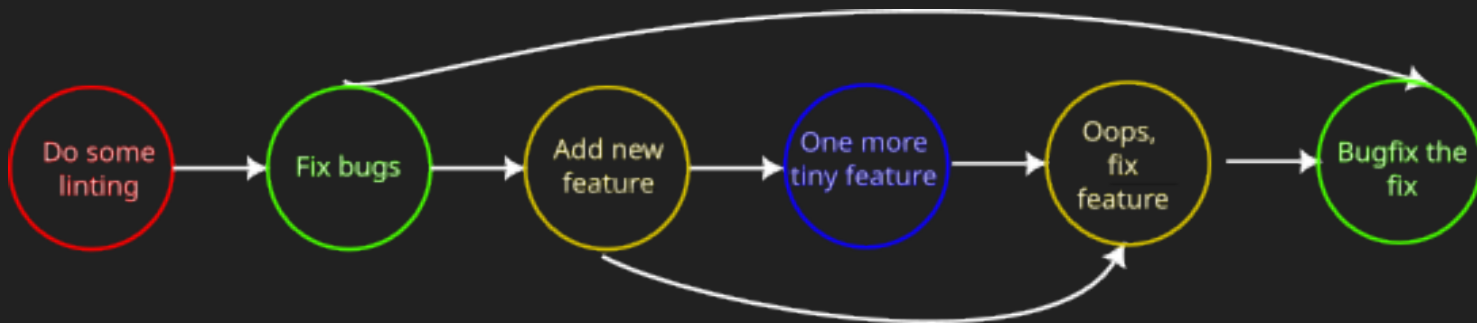
- Pull request — the unit of review
- Discussion happens – more commits added on top



Code review today

Most common method popularised by Github

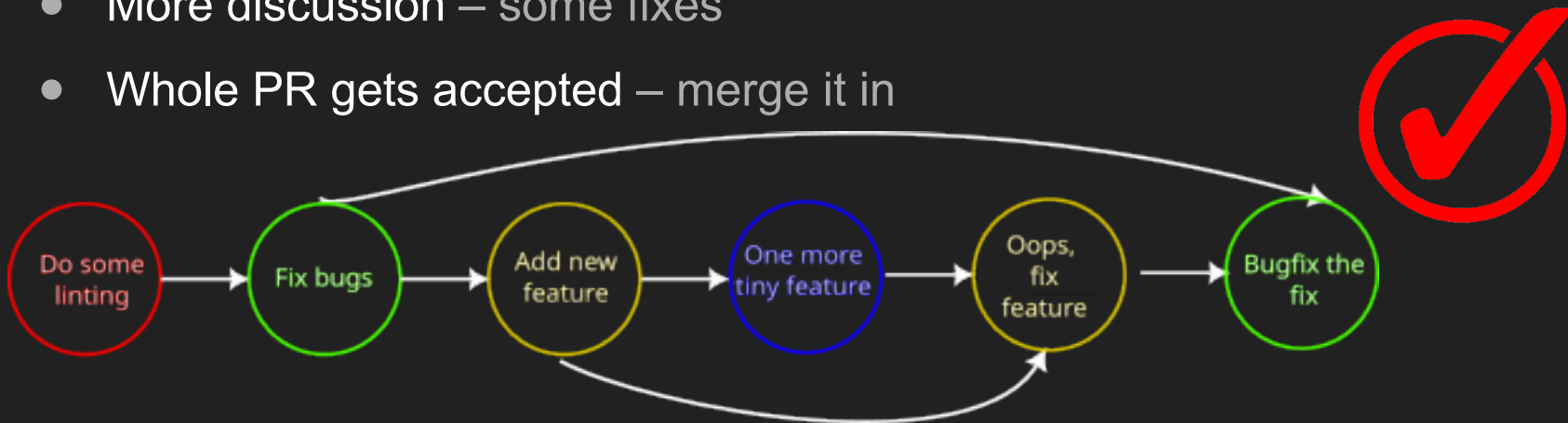
- Pull request — the unit of review
- Discussion happens – more commits added on top
- More discussion – some fixes



Code review today

Most common method popularised by Github

- Pull request — the unit of review
- Discussion happens – more commits added on top
- More discussion – some fixes
- Whole PR gets accepted – merge it in



Code review today

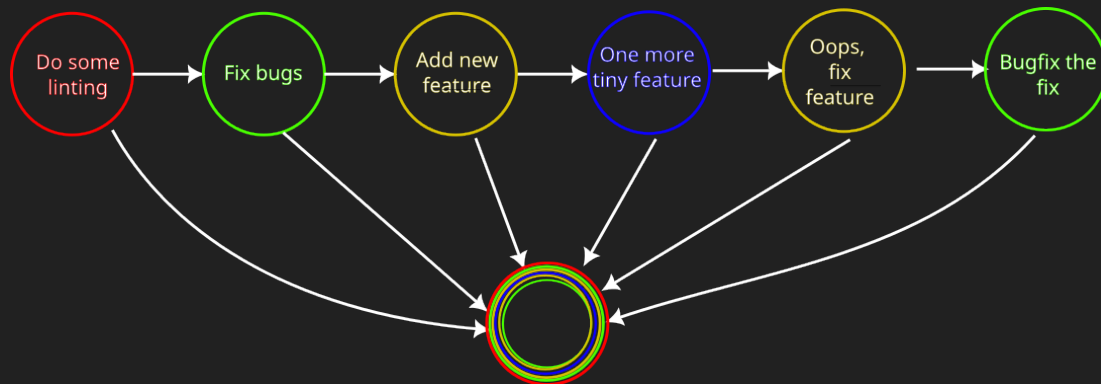
Most common method popularised by Github

- But how to merge?

Code review today

Most common method popularised by Github

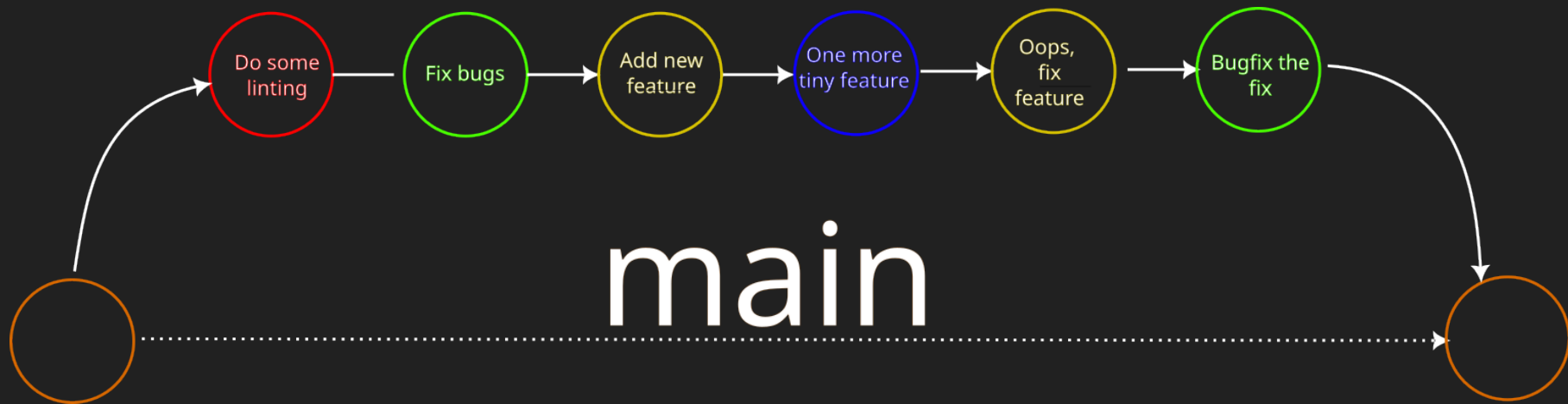
- But how to merge?
- Option 1 – squash merge
 - One big omnibus commit to history



Code review today

Most common method popularised by Github

- But how to merge?
- Option 2 – merge commit
 - Commit the whole drama to history



Code review today

Drawbacks of these merges

Code review today

Drawbacks of these merges

- Squash merge – loses logical separation

Code review today

Drawbacks of these merges

- Squash merge – loses logical separation
- Merge commit – makes history too noisy

Code review today

Drawbacks of these merges

- Squash merge – loses logical separation
- Merge commit – makes history too noisy
- Both methods

Code review today

Drawbacks of these merges

- Squash merge – loses logical separation
- Merge commit – makes history too noisy
- Both methods
 - Make bisection more difficult

Code review today

Drawbacks of these merges

- Squash merge – loses logical separation
- Merge commit – makes history too noisy
- Both methods
 - Make bisection more difficult
 - Make `git blame` more difficult

Code review today

Drawbacks of these merges

- Squash merge – loses logical separation
- Merge commit – makes history too noisy
- Both methods
 - Make bisection more difficult
 - Make `git blame` more difficult
 - Divorce commit messages from commit

Code review yesterday

Git like a kernel hacker

Code review yesterday

Git like a kernel hacker

- Mailing list – A tad byzantine but...

```
Date W      Flds  List      From      Subject
02/19/2019  S      linux-k... Peter Zijlstra + Re: [RFC][PATCH 03/16] sched: Wrap rq::lock access
02/19/2019  S      linux-k... Phil Auld      \ Re: [RFC][PATCH 03/16] sched: Wrap rq::lock access
02/18/2019  S      linux-k... Peter Zijlstra [RFC][PATCH 16/16] sched: Debug bits...
02/18/2019  S      linux-k... Peter Zijlstra [RFC][PATCH 14/16] sched/fair: Add a few assertions
02/18/2019  Nu      linux-k... Peter Zijlstra [RFC][PATCH 13/16] sched: Add core wide task selection and scheduling.
02/18/2019  Nu      linux-k... Peter Zijlstra [RFC][PATCH 12/16] sched: A quick and dirty cgroup temping interface
02/18/2019  Nu      linux-k... Peter Zijlstra [RFC][PATCH 11/16] sched: Basic tracking of matching tasks
-----
<E>  *made:headers*  322.64%  LF UTF-8  made:headers

From: Peter Zijlstra
To: mingko@kernel.org, tgla@linutronix.de, pji@google.com,
tim.c.chen@linux.intel.com, tarvalds@linux-foundation.org
Cc: linux-kernel@vger.kernel.org, subhra.mazumdar@oracle.com,
twalton@gmail.com, keescook@chromium.org, kernel@google.com, Peter Zijlstra
(Intel)
Subject: [RFC][PATCH 14/16] sched/fair: Add a few assertions
Flags: seen, list
Dates: Mon 19 Feb 2019 08:56:34 AM PST
Maildir: /INBOX
List: linux-kernel.vger.kernel.org

Signed-off-by: Peter Zijlstra (Intel) <peterz@infradead.org>
...
kernel/sched/fair.c | 12 ++++++
1 file changed, 10 insertions(+), 2 deletions(-)

--- a/kernel/sched/fair.c
+++ b/kernel/sched/fair.c
@@ -6208,6 +6208,11 @@ static int select_idle_sibling(struct ta
struct sched_domain *sd;
int i, recent_used_cpu;

/*
 * per-cpu select_idle_mask usage
 */
lockdep_assert_irqs_disabled();
if (available_idle_cpu(target))
return target;

@@ -6635,8 +6640,6 @@ static int find_energy_efficient_cpu(str
/* certain conditions an idle sibling CPU if the domain has SD_WAKE_AFFINE set.
...
<E>  *made:view*  1 Top  LF UTF-8  made:view
```

```
/*
 * Wake up if we have enough events, or if a timeout occurred since we
 * started waiting. For timeouts, we always want to return to userspace,
 * regardless of event count.
 */
return dist >= 0 || atomic_read(&ctx->cq_timeouts) != iowq->nr_timeouts;
}
```

```
static int io_wake_function(struct wait_queue_entry *curr, unsigned int mode,
int wake_flags, void *key)
```

```
{
    iff --git a/io_uring/io_uring.h b/io_uring/io_uring.h
index 25515d69d205..5f04bd47562a 100644
--- a/io_uring/io_uring.h
+++ b/io_uring/io_uring.h
@@ -41,6 +41,28 @@ enum {
    IOU_STOP_MULTISHOT = -ECANCELED,
};
```

```
struct io_wait_queue {
    struct wait_queue_entry wq;
    struct io_ring_ctx *ctx;
    unsigned cq_tail;
    unsigned nr_timeouts;
    ktime_t timeout;
};
```

Code review yesterday

Git like a kernel hacker

- Mailing list – A tad byzantine but...
 - Makes the commit the unit of review

Code review yesterday

Git like a kernel hacker

- Mailing list – A tad byzantine but...
 - Makes the commit the unit of review
 - Allows accepting a subset of a patch series

Code review yesterday

Git like a kernel hacker

- Mailing list – A tad byzantine but...
 - Makes the commit the unit of review
 - Allows accepting a subset of a patch series
 - Encourages full commit messages


Code review yesterday

Git like a kernel hacker

- Mailing list – A tad byzantine but...
 - Makes the commit the unit of review
 - Allows accepting a subset of a patch series
 - Encourages full commit messages
 - Amends are just another email

But then again...

But then again...

... does it really have
to be email? 

Nope!



Another world is possible

Un autre monde est possible

Another world is possible

Un autre monde est possible

- Better tools for writing commits

Another world is possible

Un autre monde est possible

- Better tools for writing commits
- Better tools for reading commits

Another world is possible

Un autre monde est possible

- Better tools for writing commits
 - jujutsu
- Better tools for reading commits

Another world is possible

Un autre monde est possible

- Better tools for writing commits
 - jujutsu
 - mercurial (with hg-git)
- Better tools for reading commits

Another world is possible

Un autre monde est possible

- Better tools for writing commits
 - jujutsu
 - mercurial (with hg-git)
 - git (with commit-editing tools)
- Better tools for reading commits

Another world is possible

Un autre monde est possible

- Better tools for writing commits
 - jujutsu
 - mercurial (with hg-git)
 - git (with commit-editing tools)
- Better tools for reading commits
 - Phorge (fork of Phabricator)

Another world is possible

Un autre monde est possible

- Better tools for writing commits
 - jujutsu
 - mercurial (with hg-git)
 - git (with commit-editing tools)
- Better tools for reading commits
 - Phorge (fork of Phabricator)
 - Review Board

Another world is possible

Un autre monde est possible

- Better tools for writing commits
 - jujutsu
 - mercurial (with hg-git)
 - git (with commit-editing tools)
- Better tools for reading commits
 - Phorge (fork of Phabricator)
 - Review Board
 - Gerrit (use change sets)

Another world is possible

Un autre monde est possible

- Better tools for writing commits
 - jujutsu
 - mercurial (with hg-git)
 - git (with commit-editing tools)
- Better tools for reading commits
 - Phorge (fork of Phabricator)
 - Review Board
 - Gerrit (use change sets)
 - `git range-diff`

A commit log worth reading...

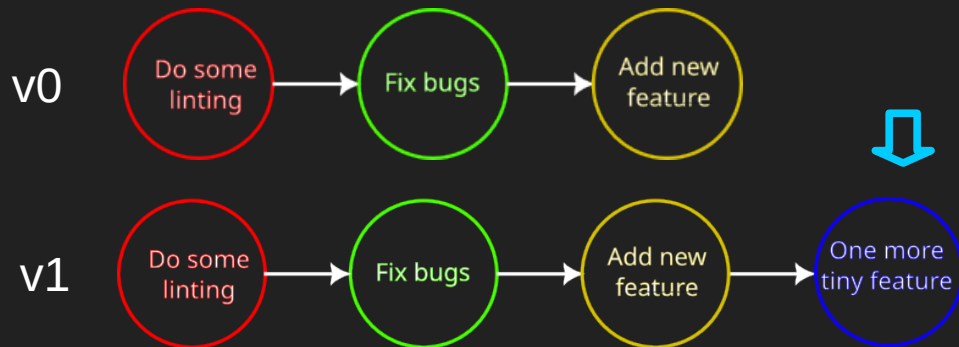
... is a commit log worth writing!

Rewrite your commits!

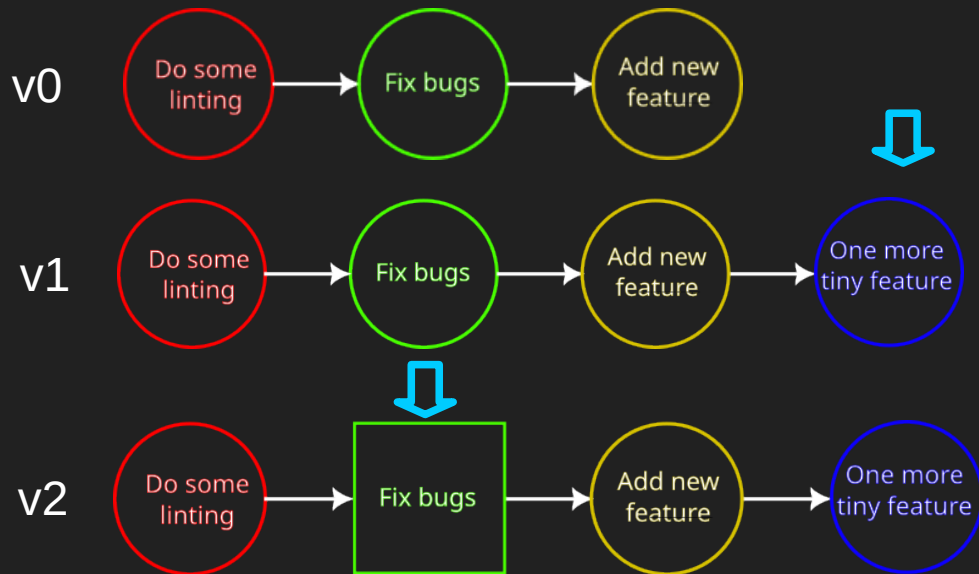
Rewrite your commits!



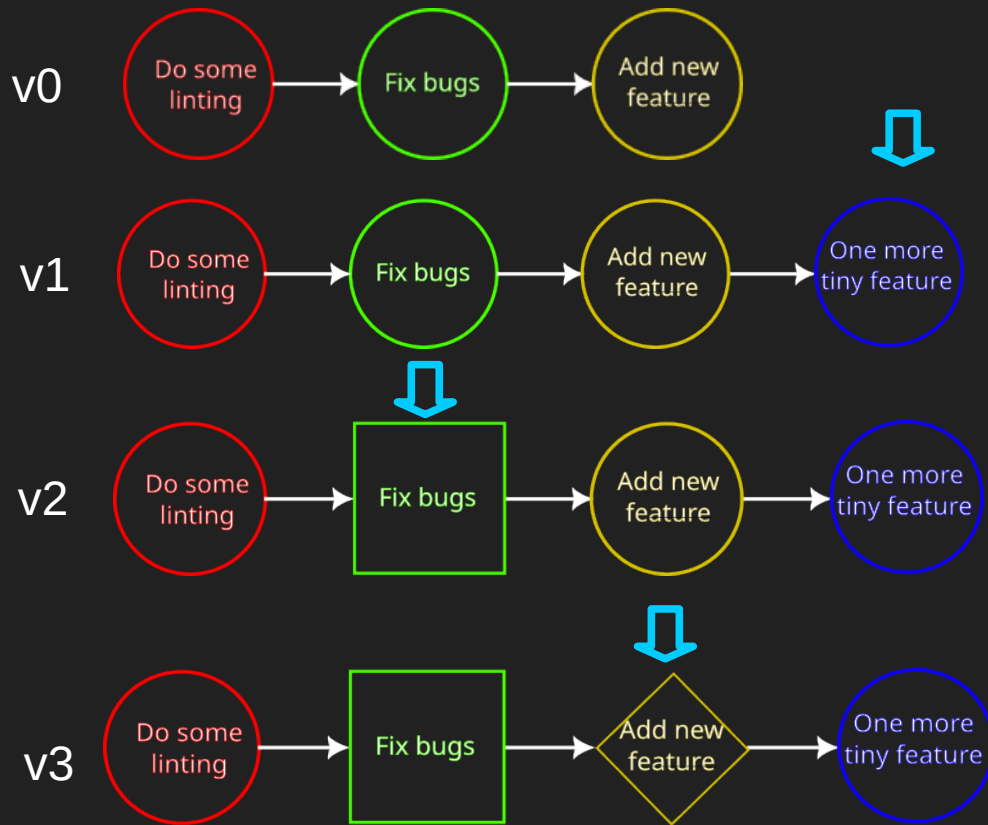
Rewrite your commits!



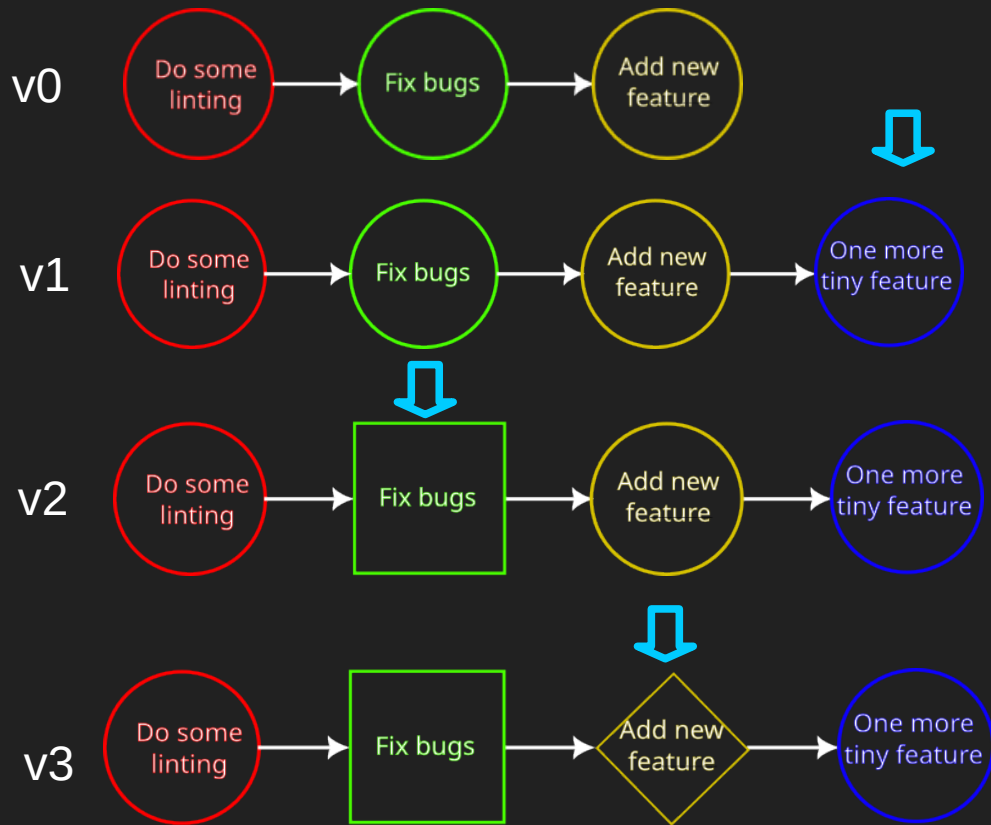
Rewrite your commits!



Rewrite your commits!



Rewrite your commits!



Commits not in main
are *always* drafts

How to write and rewrite

Each tool has its own ways

How to write and rewrite

Each tool has its own ways

- git
- mercurial
- jujutsu

How to write and rewrite

Each tool has its own ways

- git
 - `add -i`
- mercurial
 - `commit -i`
- jujutsu
 - `split`

How to write and rewrite

Each tool has its own ways

- git
 - `add -i`
 - `rebase -i`
- mercurial
 - `commit -i`
 - `histedit`
- jujutsu
 - `split`
 - GG frontend
 - jj-fzf frontend

How to write and rewrite

Each tool has its own ways

- git
 - `add -i`
 - `rebase -i`
- mercurial
 - `commit -i`
 - `histedit`
 - `evolve`
- jujutsu
 - `split`
 - GG frontend
 - jj-fzf frontend

How to write and rewrite

Each tool has its own ways

- git
 - `add -i`
 - `rebase -i`
 - other frontends
- mercurial
 - `commit -i`
 - `histedit`
 - `evolve`
- jujutsu
 - `split`
 - GG frontend
 - jj-fzf frontend

How to write and rewrite

Secret weapon common to all three

How to write and rewrite

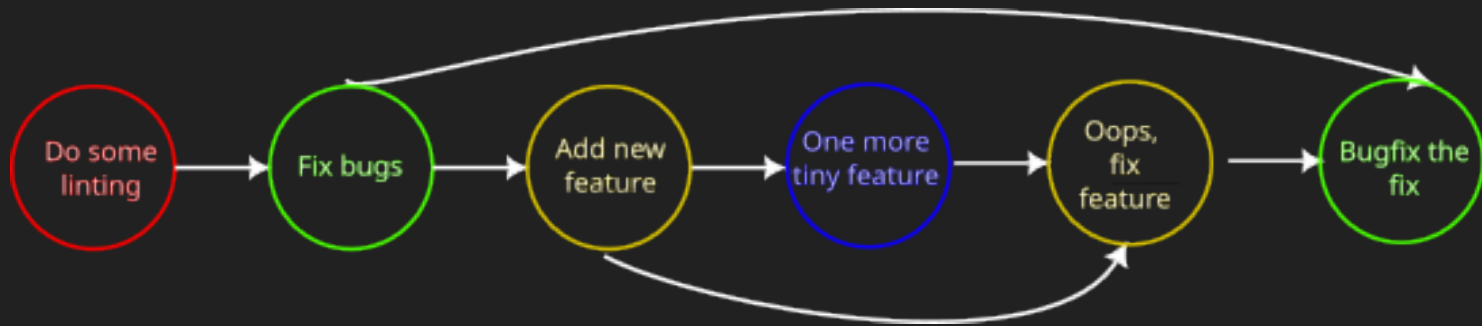
Secret weapon common to all three

git/hg/jj absorb

<https://github.com/tummychow/git-absorb>



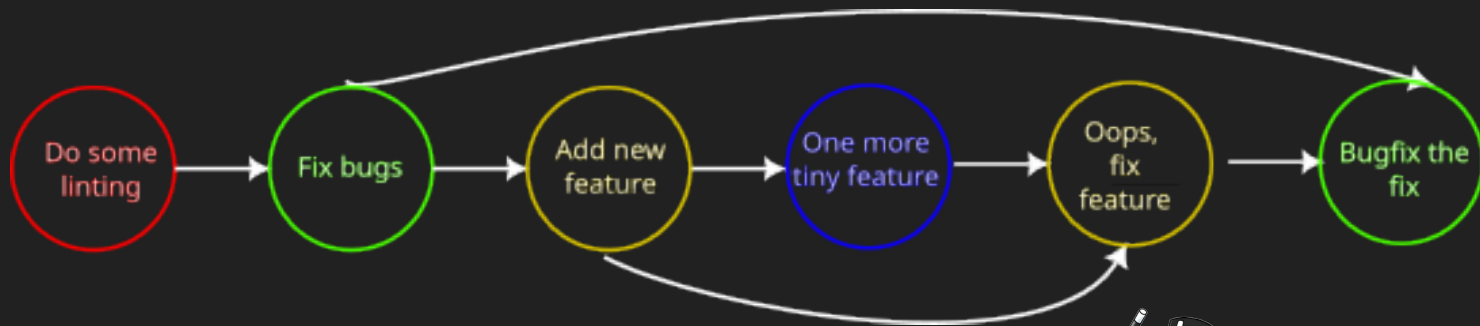
git/hg/jj absorb



<https://github.com/tummychow/git-absorb>



git/hg/jj absorb



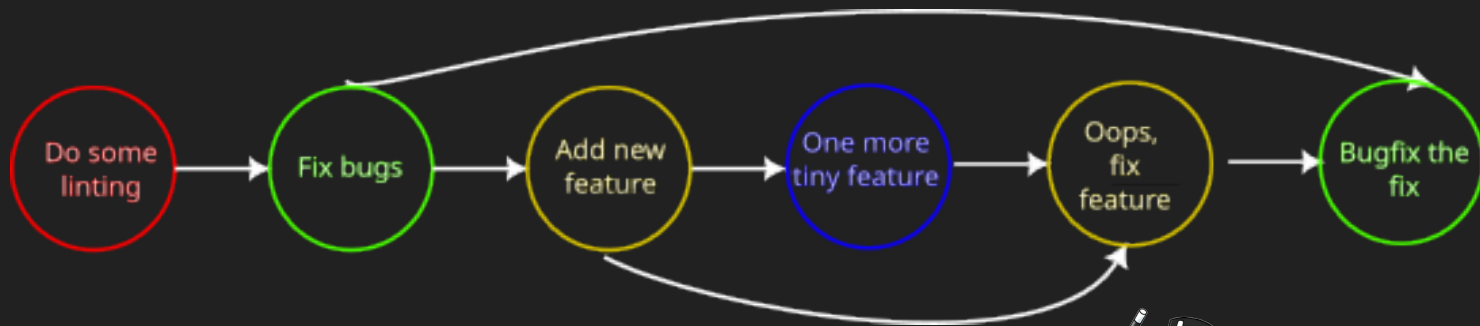
m a g i c !



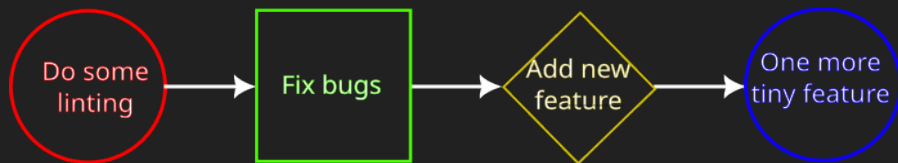
<https://github.com/tummychow/git-absorb>



git/hg/jj absorb



m a g i c !



<https://github.com/tummychow/git-absorb>

git/hg/jj absorb



<https://github.com/tummychow/git-absorb>



git/hg/jj absorb



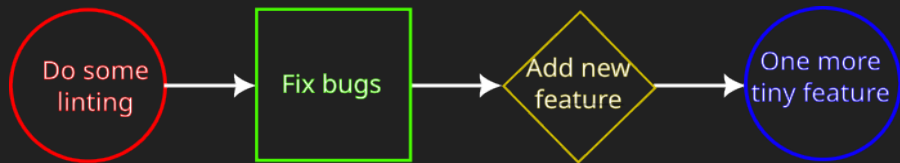
m a g i c !



<https://github.com/tummychow/git-absorb>



git/hg/jj absorb



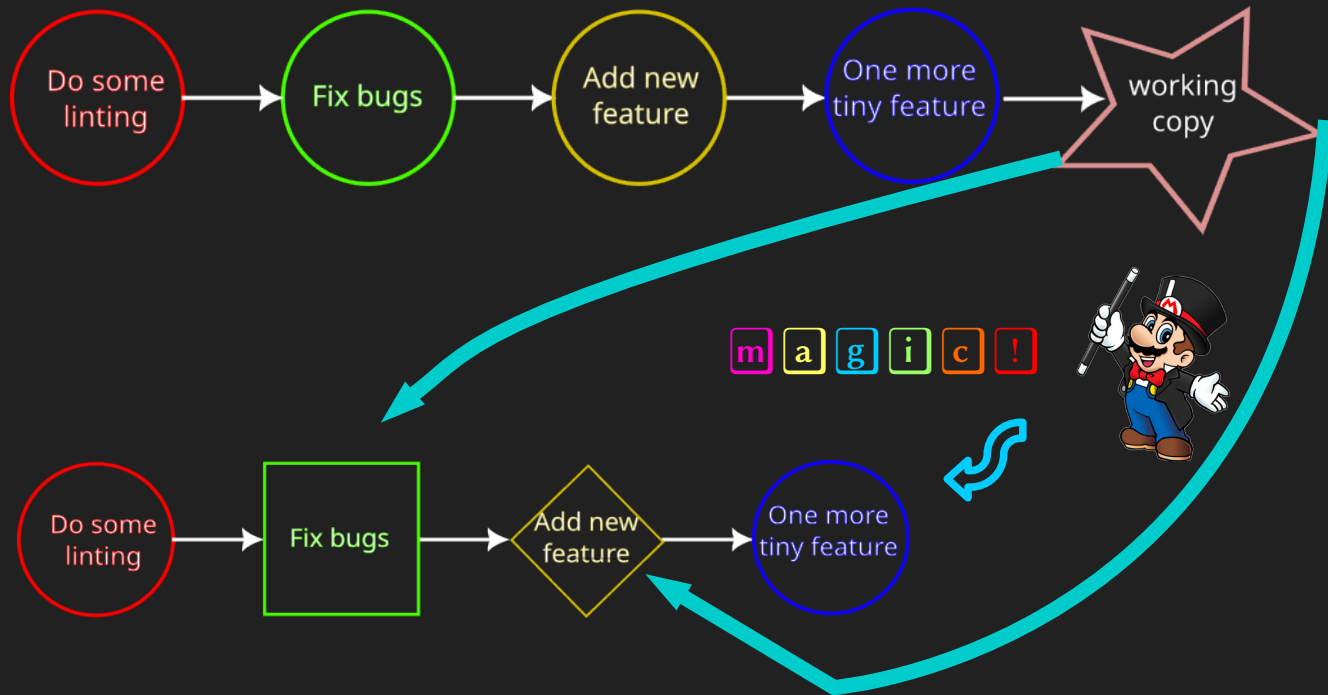
m a g i c !



<https://github.com/tummychow/git-absorb>



git/hg/jj absorb



<https://github.com/tummychow/git-absorb>



How does **absorb** work?

Fine, it's not literally magic

<https://github.com/tummychow/git-absorb>



How does **absorb** work?

Fine, it's not literally magic

- Compare working copy diff with commit stack

<https://github.com/tummychow/git-absorb>



How does **absorb** work?

Fine, it's not literally magic

- Compare working copy diff with commit stack
- Whenever a hunk from working copy overlaps...

<https://github.com/tummychow/git-absorb>



How does **absorb** work?

Fine, it's not literally magic

- Compare working copy diff with commit stack
- Whenever a hunk from working copy overlaps...
- ... associate that hunk to overlapping commit!

<https://github.com/tummychow/git-absorb>



How does **absorb** work?

Fine, it's not literally magic

- Compare working copy diff with commit stack
- Whenever a hunk from working copy overlaps...
- ... associate that hunk to overlapping commit!
- Amend each commit with its associated hunks

<https://github.com/tummychow/git-absorb>



How to write and rewrite

An important concept (used by jujutsu and Gerrit)

How to write and rewrite

An important concept (used by jujutsu and Gerrit)

- A *change* is the commit as it gets edited.

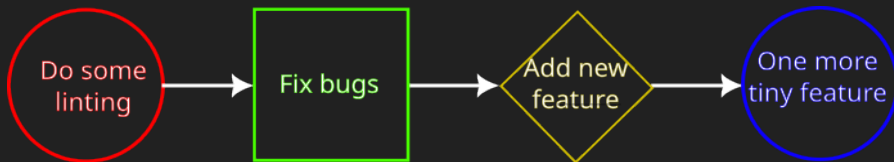
How to write and rewrite

An important concept (used by jujutsu and Gerrit)

- A *change* is the commit as it gets edited.
- There will be many different commit hashes, but several commits may have the same change ID.

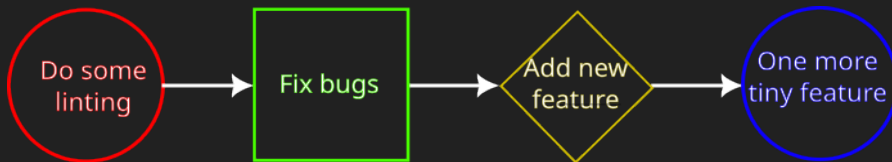
How to write and rewrite

An important concept (used by jujutsu and Gerrit)



How to write and rewrite

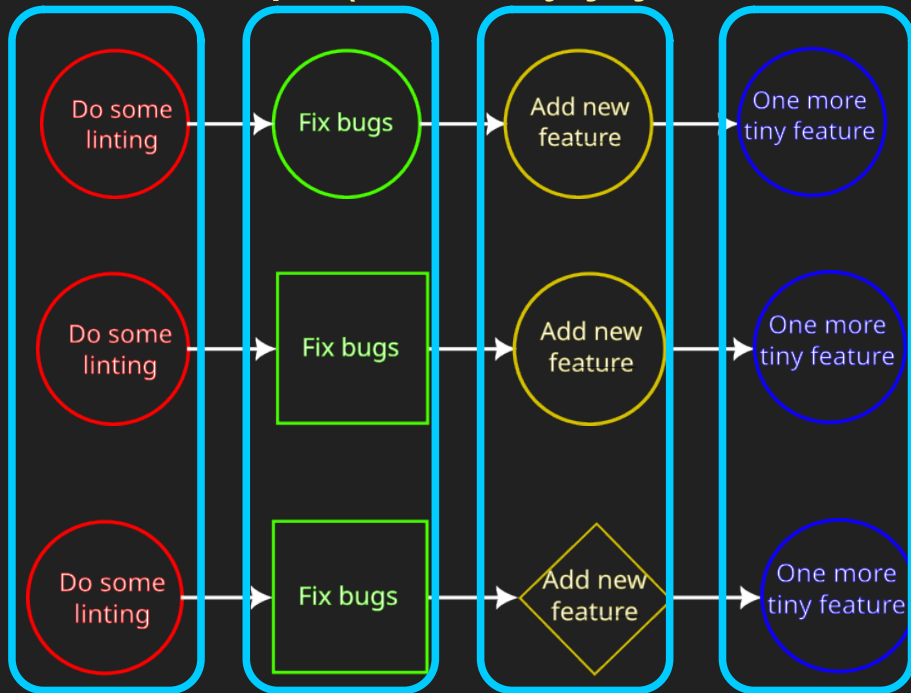
An important concept (used by jujutsu and Gerrit)



- 12 commits but

How to write and rewrite

An important concept (used by jujutsu and Gerrit)



- 12 commits but
- only 4 changes

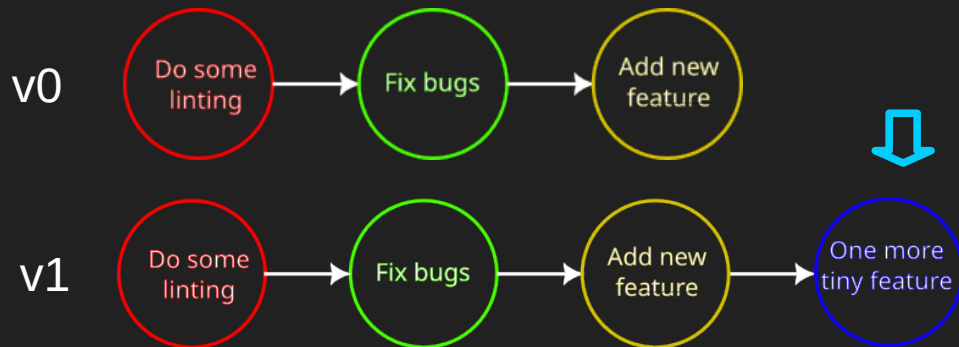
Now let's talk about
how to read commits

Review their commits

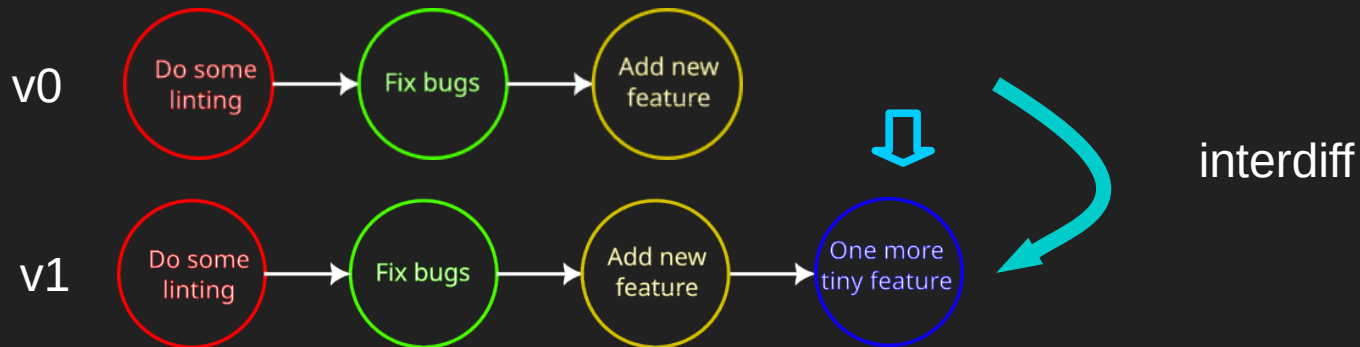
Review their commits



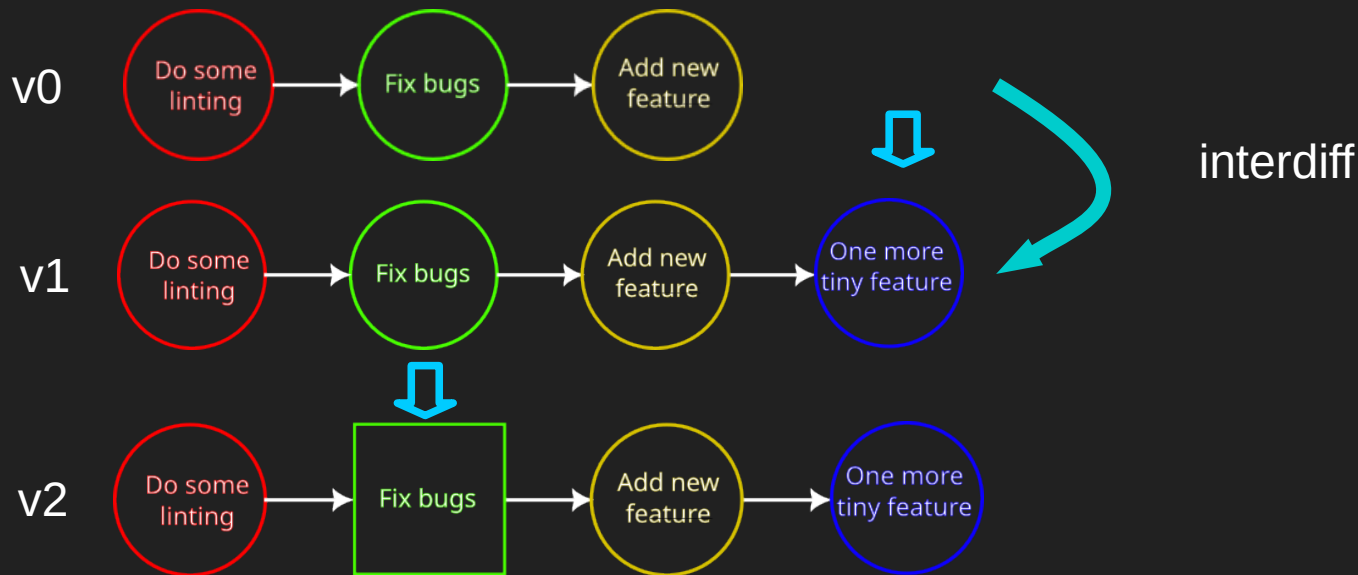
Review their commits



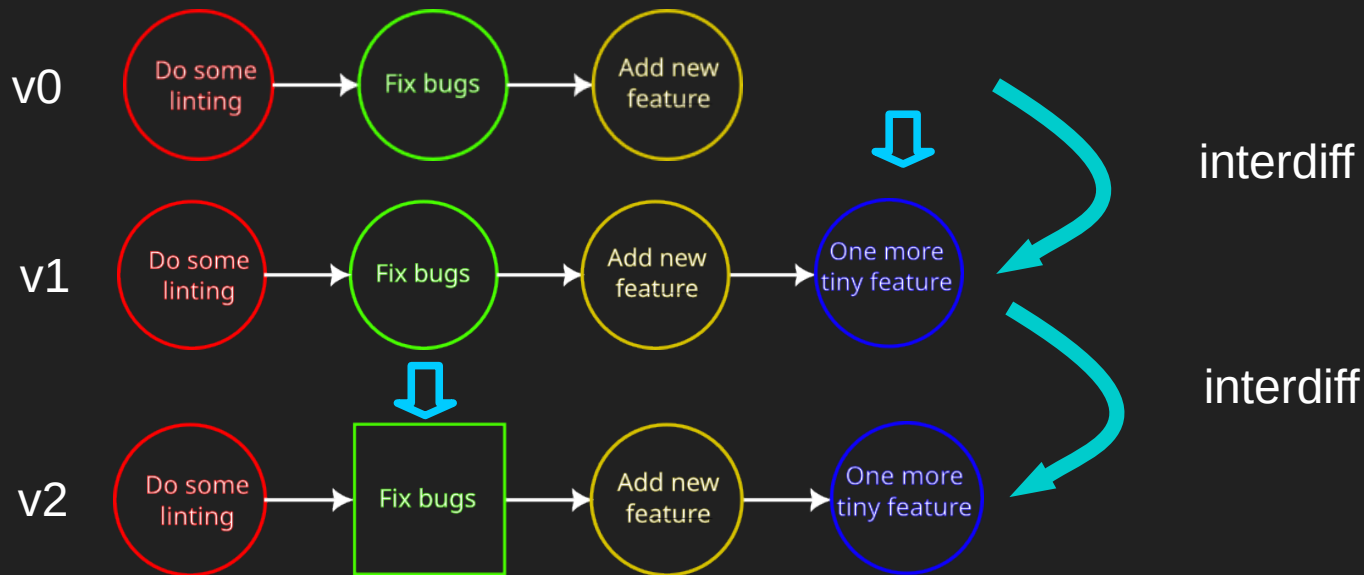
Review their commits



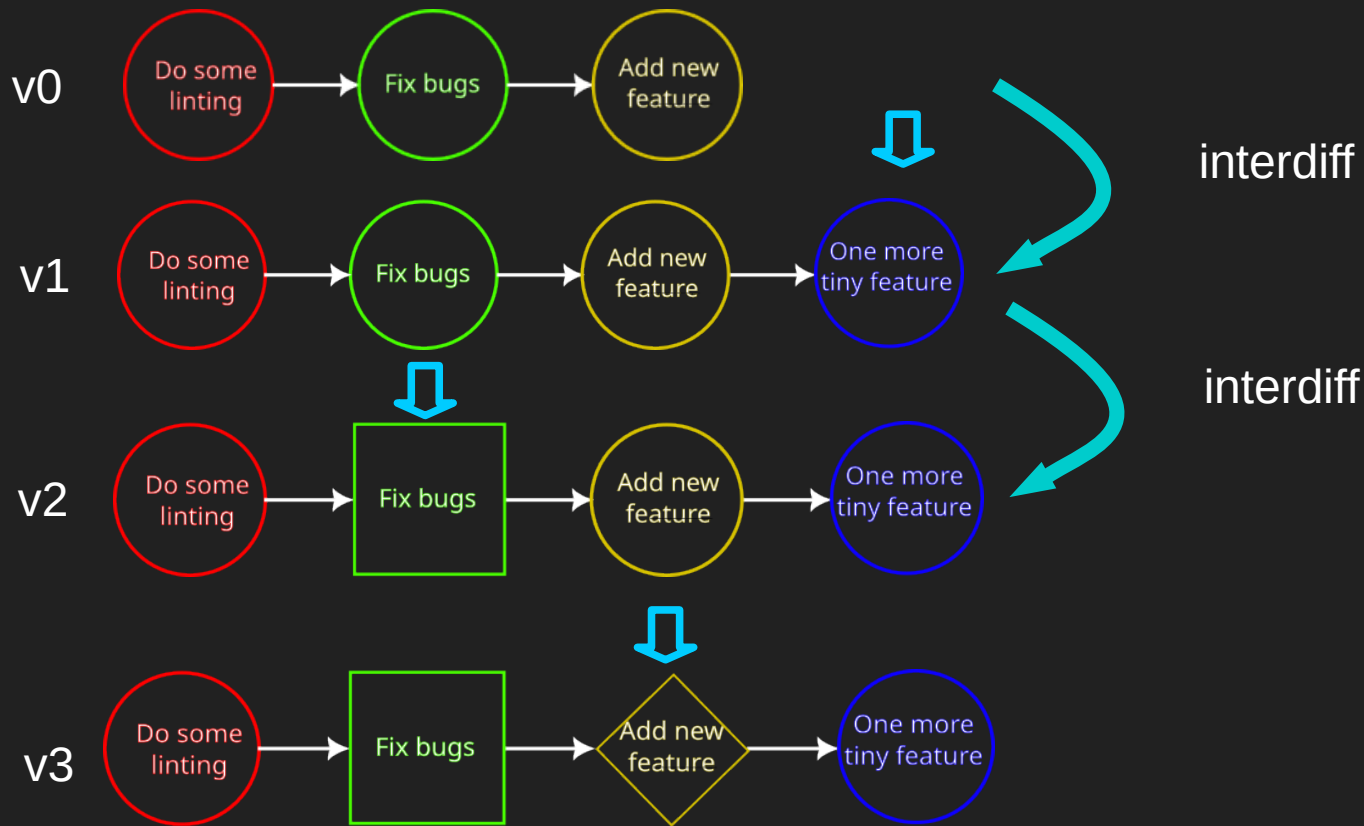
Review their commits



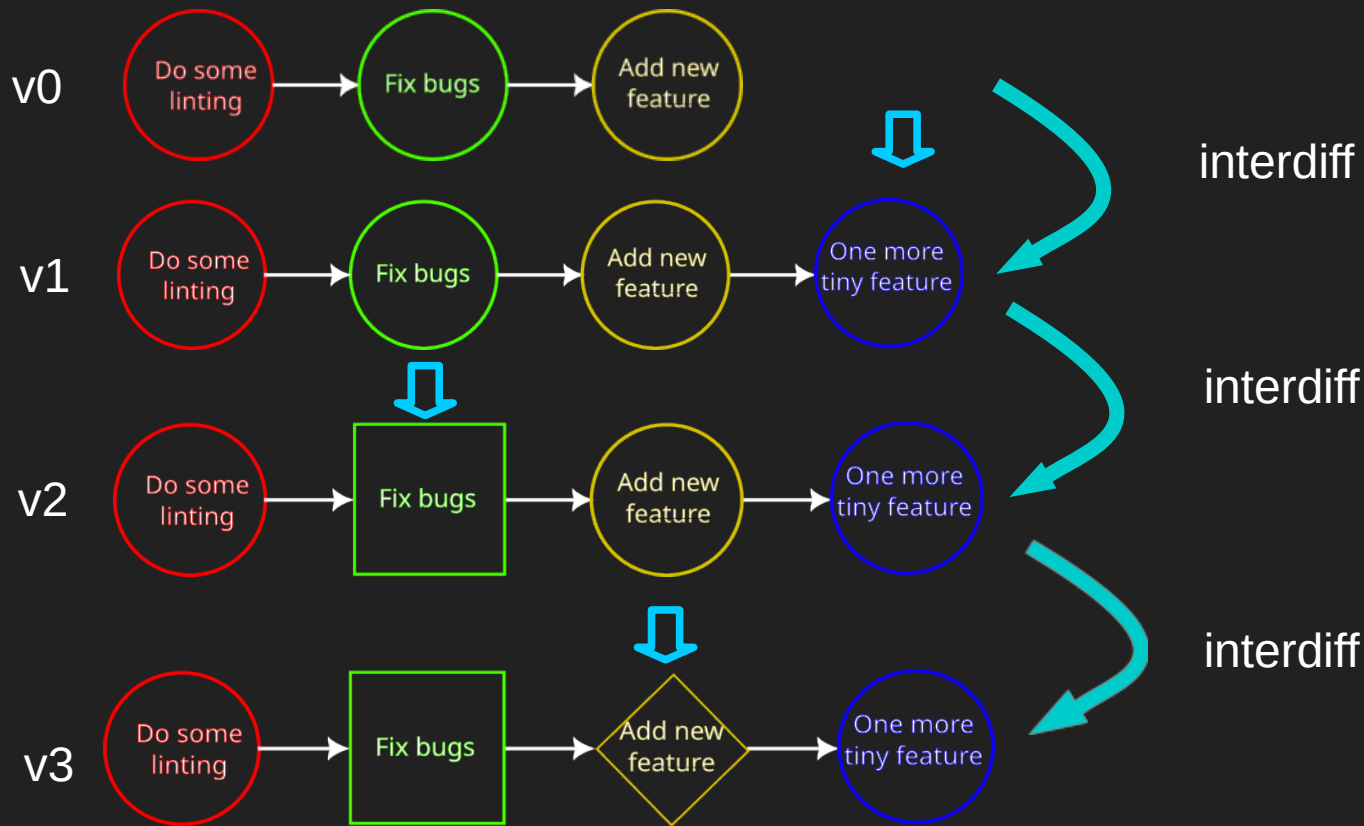
Review their commits



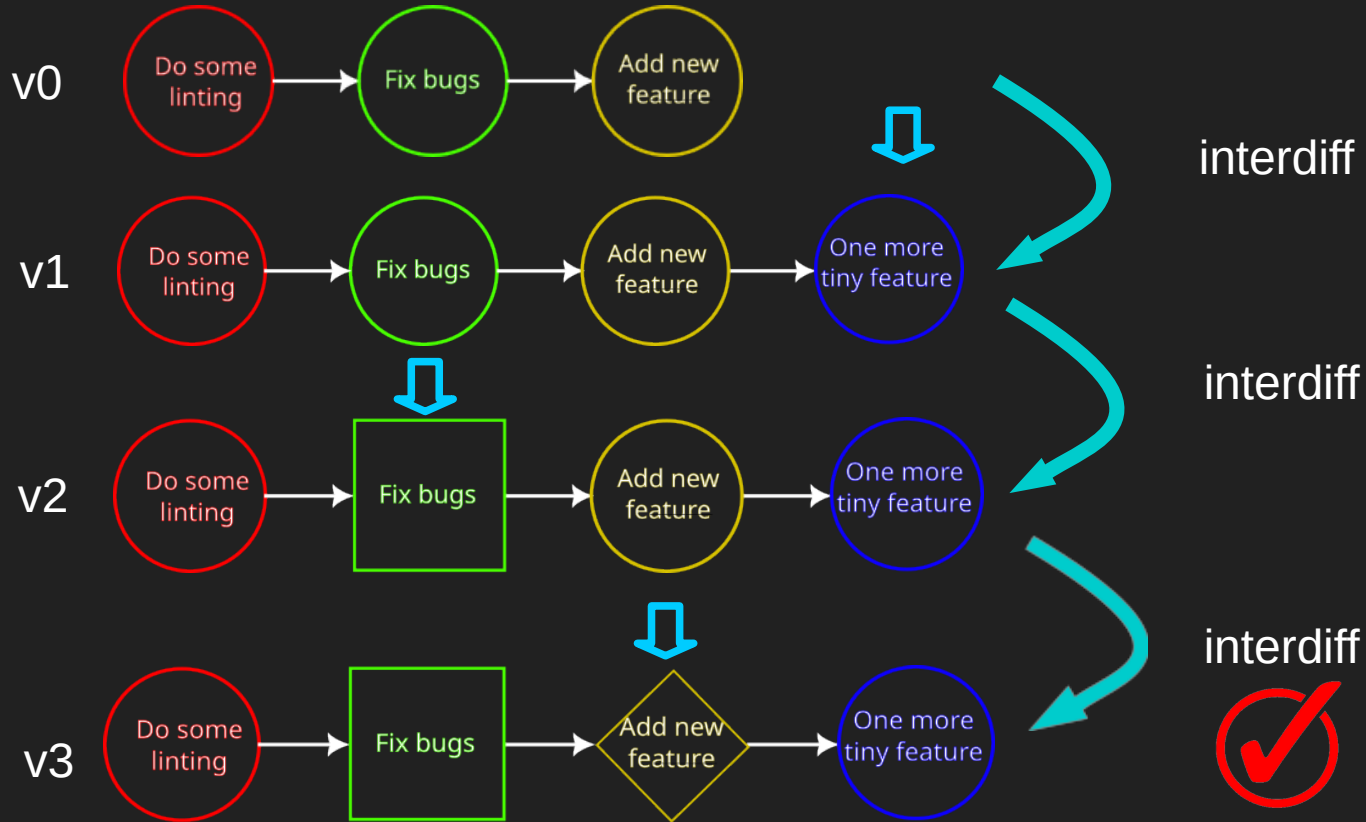
Review their commits



Review their commits



Review their commits



Interdiffs

What are they?

Interdiffs

What are they?

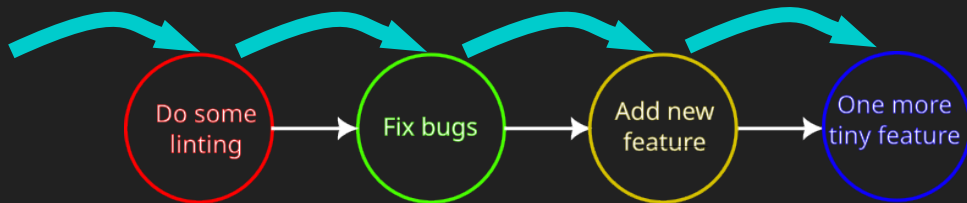
- Commits are *snapshots*



Interdiffs

What are they?

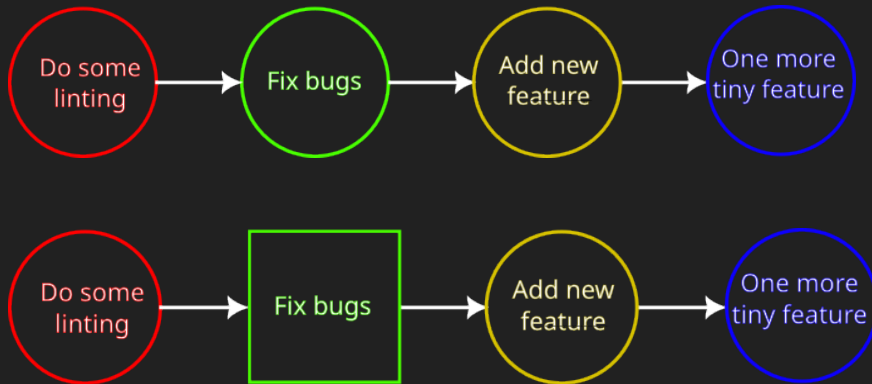
- Commits are *snapshots*
- So you can diff them like this



Interdiffs

What are they?

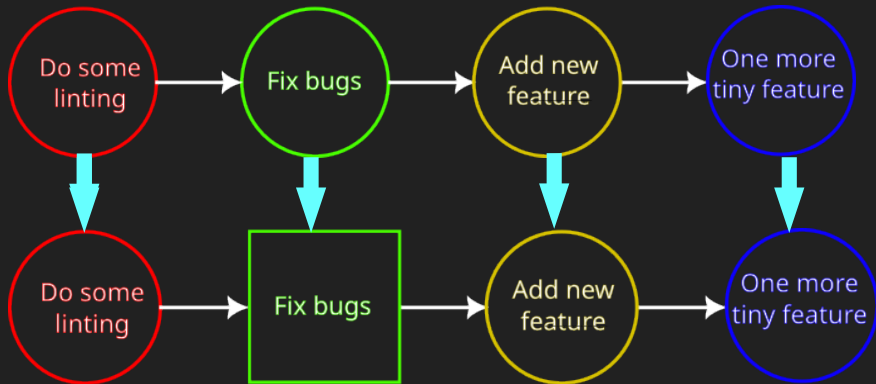
- Commits are *snapshots*
- So you can diff them like this
- But with two sequences of commits...



Interdiffs

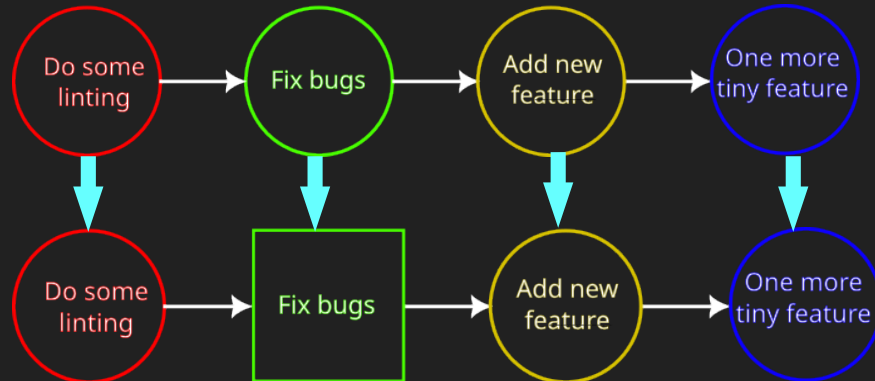
What are they?

- Commits are *snapshots*
- So you can diff them like this
- But with two sequences of commits...
- You can diff them like this!



Interdiffs

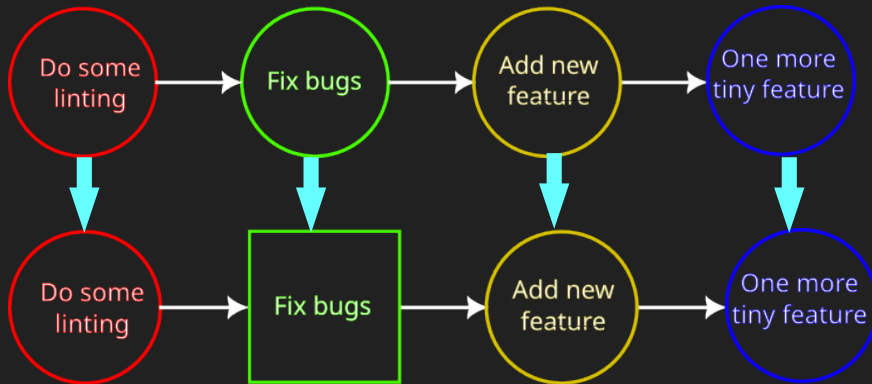
What are they?



Interdiffs

What are they?

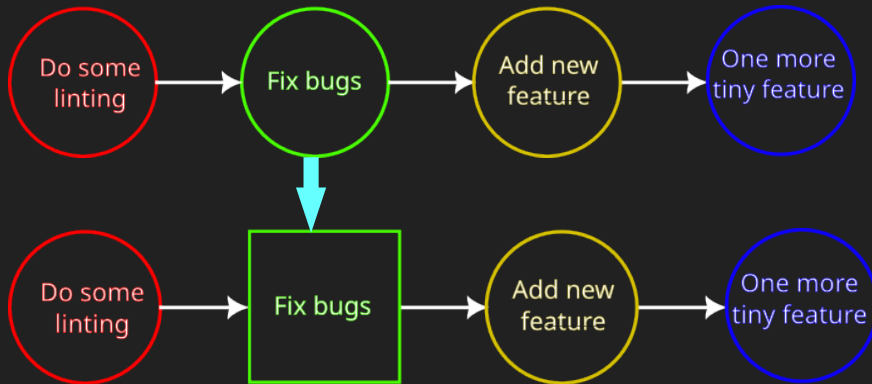
- Since most commits are the same



Interdiffs

What are they?

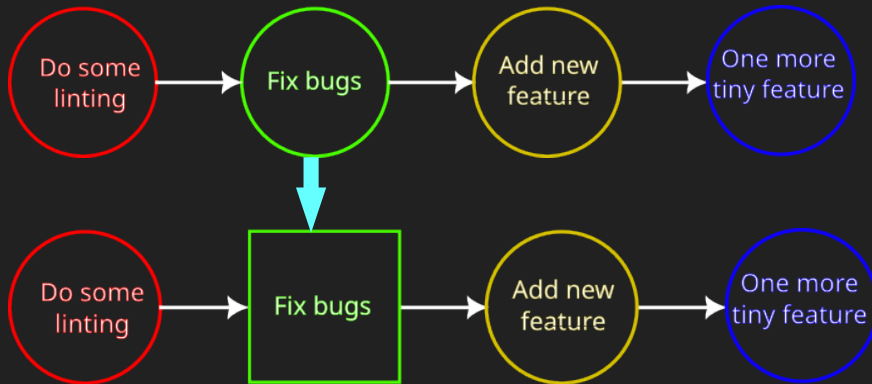
- Since most commits are the same
- The interdiff just leaves one real difference



Interdiffs

What are they?

- Since most commits are the same
- The interdiff just leaves one real difference
- Much easier to review!



How to read and review

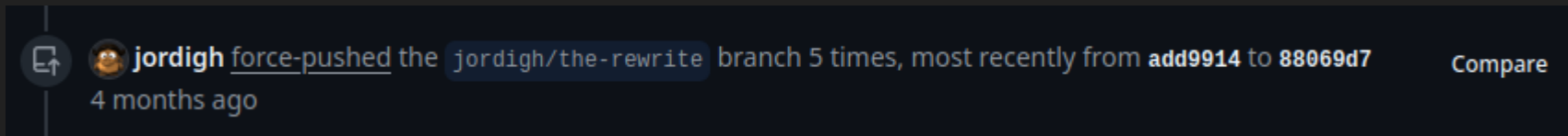
How to read and review

Unfortunately, Github doesn't offer a lot of help here

How to read and review

Unfortunately, Github doesn't offer a lot of help here



but there is some:



How to read and review






































Unfortunately, Github doesn't offer a lot of help here

but there is some:

  **jordigh** force-pushed the `jordigh/the-rewrite` branch 5 times, most recently from `add9914` to `88069d7`
4 months ago

Compare

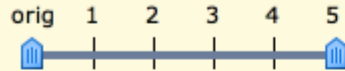
Phorge says “diffs” when it means interdiffs

Revision Contents							Changeset List
Files	History	Commits	Similar				
Diff	ID	Base	Description	Created	Lint	Unit	
Base		Base					
Diff 1	20361	e0d4450		Aug 27 2024, 9:28 PM			
Diff 2	20362	e0d4450	Fix linting	Aug 27 2024, 9:46 PM			
Diff 3	20363	e0d4450	Fix an import	Aug 27 2024, 10:32 PM			
Diff 4	20366	e0d4450	Fix a logical error.	Aug 28 2024, 10:50 AM			
Diff 5	20367	e0d4450	Fix a type-check	Aug 28 2024, 11:21 AM			
Diff 6	20368	aa694f0	Rebase onto master	Aug 28 2024, 11:46 AM			
Diff 7	20369	aa694f0	Disable deprecated lint warning	Aug 28 2024, 12:48 PM			
Diff 8	20370	aa694f0	Rename a test variable	Aug 28 2024, 4:52 PM			
Diff 9	20375	aa694f0	Address changes from code review.	Aug 29 2024, 1:37 PM			
Diff 10	20376	aa694f0	Fix logic in case there is a missing definition for the grace period	Aug 29 2024, 3:25 PM			
Diff 11	20377	aa694f0	patch up tests to set the expected grace period start	Aug 29 2024, 4:24 PM			
Diff 12	20378	fcdcd52	rG19790ce5d750152cfb4a308416a18352bc317b9d	Aug 29 2024, 10:51 PM			

How to read and review

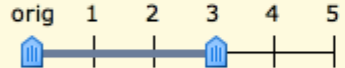
Review Board also calls interdiffs “diff revisions”

Diff Revision 5 (Latest)

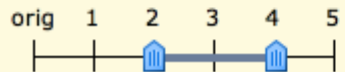


Diff Revision 3

This is not the most recent revision of the diff. The [latest diff](#) is revision 5. [See what's changed.](#)



Changes between revision 2 and 4





How to read and review

Gerrit calls similar commits “changes” (recall)
and groups them into “change sets” or “patchsets”

Submitted together

SHOW ALL (9) ▾

→ iommu/arm-smmu-v3: Rename cdcfg to cd_table	.../torvalds/linux master
iommu/arm-smmu-v3: Update comment about STE liveness	.../torvalds/linux master
▾ iommu/arm-smmu-v3: Cleanup arm_smmu_domain_finalise	.../torvalds/linux master

Patchset 16 ▾	 0811e14 
Patchset 16 0811e14	Sep 15, 2023
Patchset 15 479a2fe	Aug 31, 2023
Patchset 14 198d8d2	Aug 31, 2023
Patchset 13 dbe4271	Aug 31, 2023

How to read and review

... and there is git itself

How to read and review

... and there is git itself

```
jordi@eris:~/vcs/grist/core/.hg/git$ hg glog -Tgwip --hidden -r '4001::4022' -r '4001::4020'
```

- 3ee6e5d91a26 jordigh (2025-02-18)
 - | test: add two RelayState redirection tests
- 4b2851bdc6aa jordigh (2025-02-17)
 - | test: add a SAMLConfig test
- 0e5e555fa817 jordigh (2025-02-18)
 - | test: add two RelayState redirection tests
- 3a0b5695d2da jordigh (2025-02-17)
 - | test: add a SAMLConfig test
- c53838b2344f jordigh (2025-02-03)
 - | SAMLConfig: defer the initial logic to _processInitialRequest
- 5b4dd0652d3b jordigh (2025-01-31)
 - | SAMLConfig: new utility function, _processInitialRequest
- 267c88c2695e jordigh (2025-01-31)
 - | SAMLConfig: new utility function, checkRedirectUrl
- e21713d33172 jordigh (2025-02-03)
 - | ScopedSession: default to request session instead of empty object
- 5c64683b93f0 paulfitz (2025-02-17)
 - | (core) updates from grist-core

How to read and review

... and there is git itself

```
jordi@eris:~/vcs/grist/core/.hg/git$ git range-diff 5c64683b..0e5e555fa 5c64683b..3ee6e5d91
1: e21713d3 = 1: e21713d3 ScopedSession: default to request session instead of empty object
2: 267c88c2 = 2: 267c88c2 SamlConfig: new utility function, checkRedirectUrl
3: 5b4dd065 = 3: 5b4dd065 SamlConfig: new utility function, _processInitialRequest
4: c53838b2 = 4: c53838b2 SamlConfig: defer the initial logic to _processInitialRequest
5: 3a0b5695 ! 5: 4b2851bd test: add a SAMLConfig test
@@ test/server/lib/SamlConfig.ts (new)
+   const loginSamlPath = path.resolve(testUtils.fixturesRoot, 'saml/saml-login');
+   const logoutSamlPath = path.resolve(testUtils.fixturesRoot, 'saml/saml-logout');
+
-+   testUtils.setTmpLogLevel('error');
-+
+   const setupTestServer = (enabled: boolean) => {
+     let homeUrl: string;
+     let oldEnv: testUtils.EnvironmentSnapshot;
6: 0e5e555f = 6: 3ee6e5d9 test: add_two RelayState redirection tests
```

The benefits

Code review is more effective

The benefits

Code review is more effective

- Reviewers can review individual commits

The benefits

Code review is more effective

- Reviewers can review individual commits
 - smaller units of review means better attention spans

The benefits

Code review is more effective

- Reviewers can review individual commits
 - smaller units of review means better attention spans
- New revisions of the PR/patch set are clearer

The benefits

Code review is more effective

- Reviewers can review individual commits
 - smaller units of review means better attention spans
- New revisions of the PR/patch set are clearer
 - the differences are kept localised to each change

The benefits

Commits are more informative

The benefits

Commits are more informative

- Commit messages convey pertinent information

The benefits

Commits are more informative

- Commit messages convey pertinent information
- Not this


	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

The benefits

Commits are more informative

- Commit messages convey pertinent information
- Not this
- But this



```
fs: add vfs_parse_fs_param_source() helper
Add a simple helper that filesystems can use in their parameter parser
to parse the "source" parameter. A few places open-coded this function
and that already caused a bug in the cgroup v1 parser that we fixed.
Let's make it harder to get this wrong by introducing a helper which
performs all necessary checks.

Link: https://syzkaller.appspot.com/bug?id=6312526aba5beae046fdae8f00399f87aab48b12
Cc: Christoph Hellwig <hch@lst.de>
Cc: Alexander Viro <viro@zeniv.linux.org.uk>
Cc: Dmitry Vyukov <dvyukov@google.com>
Signed-off-by: Christian Brauner <christian.brauner@ubuntu.com>
Signed-off-by: Linus Torvalds <torvalds@linux-foundation.org>
```

The benefits

`git blame` becomes source-level documentation

The benefits

`git blame` becomes source-level documentation

- every line of source code now has a good commit message

The benefits

`git blame` becomes source-level documentation

- every line of source code now has a good commit message
- want to know why this line of code is so weird?

The benefits

`git blame` becomes source-level documentation

- every line of source code now has a good commit message
- want to know why this line of code is so weird?
 - check its commit message!

The benefits

bisection works much better

The benefits

bisection works much better

- commits are atomic and single-purpose so, therefore

The benefits

bisection works much better

- commits are atomic and single-purpose so, therefore
- when you run a bisection, you know *exactly* where the bug is

The benefits

Getting good changes faster (smaller commits, but sooner)

Interdiff review systems typically encourage smaller, more "juicy" patches that land in the main branch more quickly than the alternative. You don't have to wait on all 5 commits to be ready to go; maybe the first 3 are OK, and the last 2 need more work. You merge 3 out of 5.

- Austin Seipp, *Why some of us like "interdiff" code review*

Thank you!

Email: jordi@getgrist.com

Fediverse: [@jordigh@mathstodon.xyz](https://mathstodon.xyz/@jordigh)

Thank you!

Email: jordi@getgrist.com

Fediverse: [@jordigh@mathstodon.xzy](https://mathstodon.xyz/@jordigh)

(seriously, try [absorb](#))

