

Collaborating on Patch Series

Josh Triplett
josh@joshtriplett.org

Linux Plumbers Conference 2016

Conway's Law

Conway's Law

“organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations”

(If you have four groups working on a compiler, you'll get a 4-pass compiler.)

We have thousands of developers.

We have thousands of developers.

Maintainers aggregate patches
from many developers.

We have thousands of developers.

Maintainers aggregate patches
from many developers.

The average patch series has one developer.

Our tools encode our processes

Our tools encode our processes

Our processes affect our software structure

One logical change per patch.

One logical change per patch.

The software should build and work
after each patch.

How does Git enable those processes?

Staging area

```
git add -p  
git reset -p  
git stash
```

Many tools to rewrite history

```
git commit --amend
```

```
git rebase -i
```

```
git cherry-pick
```

Developing a series of patches over time

Developing a series of patches over time

```
git rebase -i
```


Collaboration

Collaboration

vs

“Don’t rebase published history”

Git tracks history

We rewrite history

We need the history of history

Common solutions

git submodule

git submodule

Separated from original repository

git submodule

Separated from original repository

Complex additional configuration for fetch

git submodule

Separated from original repository

Complex additional configuration for fetch

Tracks commit hash, not patch series

git submodule

Separated from original repository

Complex additional configuration for fetch

Tracks commit hash, not patch series

Not enough metadata - no base

git submodule

Doesn't help with collaboration
if submodule history rewritten

Pull the patches out of git

Pull the patches out of git

quilt

Pull the patches out of git

quilt

debian/patches

Pull the history of the patch series out of git

Pull the history of the patch series out of git

Versioned branch names

Copy patches around (email or git)

Copy patches around (email or git)

Serialized collaboration model

Develop using central repository
then refactor into patch series

Develop using central repository
then refactor into patch series

Rebasing or submitting becomes an ordeal

Develop using central repository
then refactor into patch series

Rebasing or submitting becomes an ordeal
(Usually done by one person)

Develop using central repository
then refactor into patch series

Rebasing or submitting becomes an ordeal
(Usually done by one person)

Disconnects devs from logical patch series

git-series

`git-series`

Manage both the patches and history in git

Storage format critical
Incrementally improve tools

Storage format critical
Incrementally improve tools

INTERNALS.md

✓ Storage format and plumbing

- ✓ Storage format and plumbing
- ✓ Logs, patch generation

- ✓ Storage format and plumbing
- ✓ Logs, patch generation
- ✓ Push and pull

- ✓ Storage format and plumbing
- ✓ Logs, patch generation
- ✓ Push and pull
- ✓ Diffs (demo!)

- ✓ Storage format and plumbing
- ✓ Logs, patch generation
- ✓ Push and pull
- ✓ Diffs (demo!)
- ✗ Merges

What's hard in your workflow today?

What's hard in your workflow today?

How else can we enable collaboration
on patch series?

What's hard in your workflow today?

How else can we enable collaboration
on patch series?

What should core git handle natively?