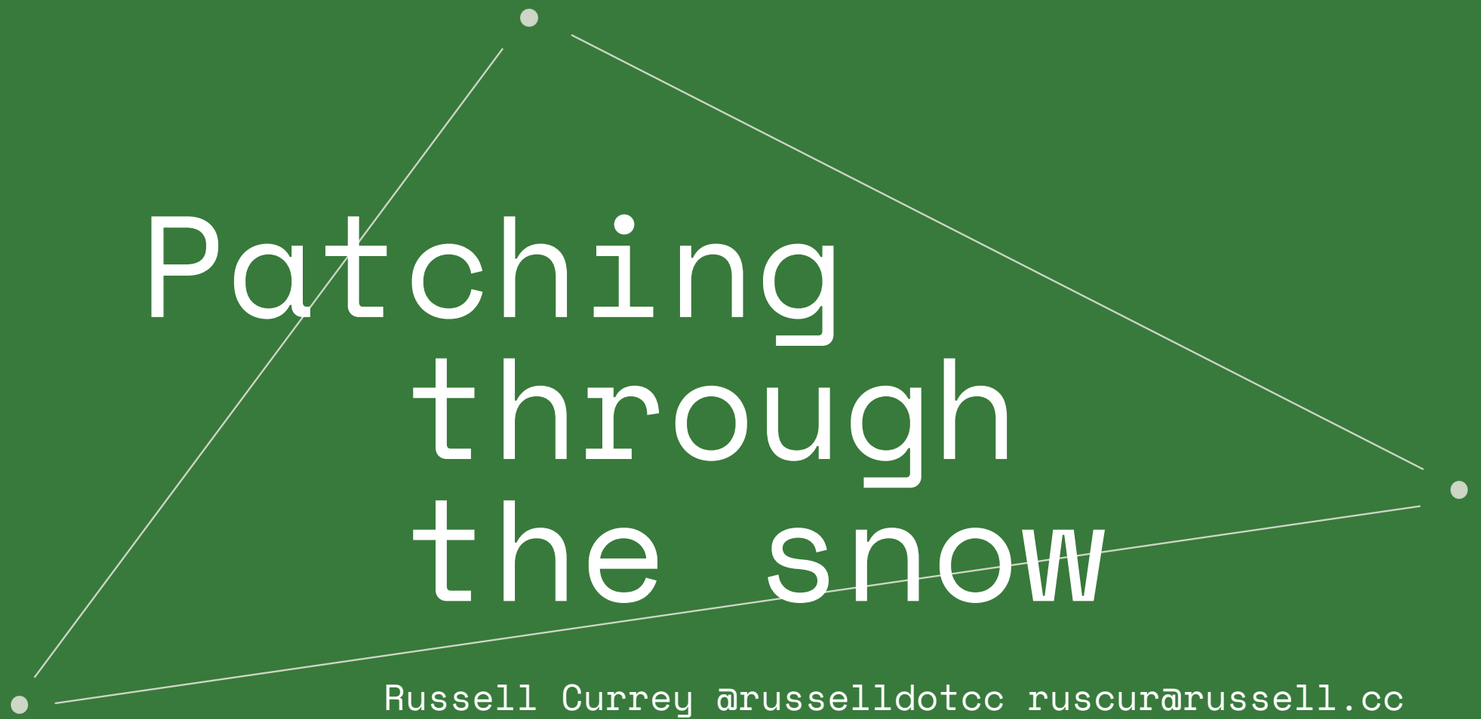

Patching through the snow

Russell Currey @russelldotcc ruscur@russell.cc

LINUX.CONF.AU
21-25 January
2019
Christchurch, NZ

Hi Christchurch!

- I’m Russell Currey
- Kernel hacker @ IBM at OzLabs in Canberra

- Kernel continuous integration
- snowpatch
- github.com/ruscur/snowpatch

What is continuous integration?

“Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control.” - Microsoft

What is continuous integration?

“Continuous Integration (CI) at its core is about feedback. It’s about implementing a process that facilitates feedback so that problems can be identified and corrected early in the development process - when it is easier to diagnose and correct them.” - CollabNet

The value of feedback

- Developers fix their code before anyone has to spend time telling them what they need to fix
- Reviewers save time reviewing what automation can catch
- Maintainers save time and have a quick sanity check
- The whole code submission process gets a lot faster

What is continuous integration… for the kernel?

- Merge patches as soon as they hit mailing lists
- Run tests, ranging from light to heavy:
  - Quick checks like checkpatch
  - Builds, possibly for multiple configs, platforms, arches
  - Tests, from selftests to userspace
- Report results

What makes this difficult?

A pull request on GitHub is a lot easier to work with than an email sent to a mailing list.

What makes this difficult?

```
Content-Type: text/plain; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [v13,07/10] powerpc: 'current_set' is now a table of task_struct pointers
Message-Id: <7e70814b903a6712b74ff2648ce2fc927558a9d8.1547195978.git.christophe.leroy@c-s.fr>
To: Benjamin Herrenschmidt <benh@kernel.crashing.org>,
 Paul Mackerras <paulus@samba.org>, Michael Ellerman <mpe@ellerman.id.au>,
 Nicholas Piggin <npiggin@gmail.com>, Mike Rapoport <rppt@linux.ibm.com>
Cc: linuxppc-dev@lists.ozlabs.org, linux-kernel@vger.kernel.org
Date: Sat, 12 Jan 2019 09:55:46 +0000 (UTC)
From: Christophe Leroy <christophe.leroy@c-s.fr>
List-Id: Linux on PowerPC Developers Mail List <linuxppc-dev.lists.ozlabs.org>
```

What makes this difficult?

Modify some parser diagnostics to continue evaluating beyond the parser #57540

Open **estebank** wants to merge 15 commits into `rust-lang:master` from `estebank:eval-more`

Conversation 15 | Commits 15 | Checks 1 | Files changed 41 | +344 −141

estebank commented 2 days ago — Contributor

Continue evaluating further errors after parser errors on:

- trailing type argument attribute
- lifetime in incorrect location
- incorrect binary literal
- missing `for` in `impl Trait for Foo`
- type argument in `where` clause

Reviewers

petrochenkov

Assignees

petrochenkov

Labels

S-waiting-on-bors

What makes this difficult?

Mailing lists have a massive lack of *metadata*.

It's hard to know if something has been merged, or rejected, or superseded.

Who's already doing kernel CI?

- Intel's 0-Day CI service
- kernelci.org (on trees, not on patches)
- snowpatch for linuxppc-dev
- Probably lots of maintainers individually
- Probably more I'm not aware of!

Intel 0-day CI

- Tests against “more than 100 different kernel configs”
- Performs static analysis on x86
- Checks for performance regressions
- Sends email reports on failure
- Proprietary, run by a team at Intel
- Awesome, but you can’t get it to do what you want
- How do maintainers know tests have been run?

So how can we bridge the gap?

- Mailing list development misses a ton of metadata that GitHub, GitLab etc have

- We can't drastically change the development workflow of the whole kernel

Patchwork!

- Web-based patch tracking system
- Patchwork supplements mailing lists, not replacing them
- Already in use by the kernel!
  - kernel.org and ozlabs.org
- Tracks patch state, series, and test results
- Has a JSON API for easy scripting

Patchwork!

Patch	Series	A/F/R/T	S/W/F	Date	Submitter	Delegate	State
[3/8] crypto4xx_core: don't abuse __dma_sync_page	[1/8] powerpc: allow NOT_COHERENT_CACHE for amigaone	- - - 1	1 1 0	2018-12-16	Christoph Hellwig		Accepted
[2/8] powerpc/dma: properly wire up the unmap_page and unmap_sg methods	[1/8] powerpc: allow NOT_COHERENT_CACHE for amigaone	- - - -	1 1 0	2018-12-16	Christoph Hellwig		Accepted
[1/8] powerpc: allow NOT_COHERENT_CACHE for amigaone	[1/8] powerpc: allow NOT_COHERENT_CACHE for amigaone	- - - -	2 0 0	2018-12-16	Christoph Hellwig		Accepted
powerpc: use mm zones more sensibly	powerpc: use mm zones more sensibly	- - - -	5 1 0	2018-12-16	Christoph Hellwig		Accepted
[RFC,6/6] powerpc: Enable support for ibm, drc-info devtree property	powerpc/pseries: Refactor code to centralize drcinfo parsing	- - - -	4 0 2	2018-12-14	Michael Bringmann		New
[RFC,5/6] powerpc/pci/hotplug: Use common drcinfo parsing	powerpc/pseries: Refactor code to centralize drcinfo parsing	- - - -	1 1 0	2018-12-14	Michael Bringmann		New
[RFC,4/6] powerpc/pseries: Use common drcinfo parsing	powerpc/pseries: Refactor code to centralize drcinfo parsing	- - - -	1 1 0	2018-12-14	Michael Bringmann		New
[RFC,3/6] pseries/drcinfo: Pseries impl of arch_find_drc_info	powerpc/pseries: Refactor code to centralize drcinfo parsing	- - - -	1 0 1	2018-12-14	Michael Bringmann		New
[RFC,2/6] pseries/drcinfo: Fix bug parsing ibm,drc-info	powerpc/pseries: Refactor code to centralize drcinfo parsing	- 1 - -	2 0 0	2018-12-14	Michael Bringmann		New
[RFC,1/6] powerpc:/drc Define interface to acquire arch-specific drc info	powerpc/pseries: Refactor code to centralize drcinfo parsing	- - - -	1 1 0	2018-12-14	Michael Bringmann		New
lkdtm: Add a tests for NULL pointer dereference	lkdtm: Add a tests for NULL pointer dereference	1 - - -	5 1 0	2018-12-14	Christophe Leroy		Not Applicable
[v2] powerpc/mm: make NULL pointer deferences explicit on bad page faults.	[v2] powerpc/mm: make NULL pointer deferences explicit on bad page faults.	- - - -	5 1 0	2018-12-14	Christophe Leroy		Accepted
[GIT,PULL] Please pull powerpc/linux.git powerpc-4.20-4 tag	[GIT,PULL] Please pull powerpc/linux.git powerpc-4.20-4 tag	- - - -	0 0 0	2018-12-14	Michael Ellerman		Not Applicable
powerpc/configs: Don't enable PPC_EARLY_DEBUG in defconfigs	powerpc/configs: Don't enable PPC_EARLY_DEBUG in defconfigs	- 1 - -	6 0 0	2018-12-14	Michael Ellerman		Accepted

powerpc: use mm zones more sensibly

Message ID	20181216165349.24281-1-hch@lst.de
State	Accepted
Commit	25078dc1f74be16b858e914f52cc8f4d03c2271a
Headers	show
Series	powerpc: use mm zones more sensibly
Related	show

Checks

Context	Check	Description
snowpatch_ozlabs/checkpatch	success	total: 0 errors, 0 warnings, 0 checks, 171 lines checked
snowpatch_ozlabs/build-pmac32	success	build succeded & removed 0 sparse warning(s)
snowpatch_ozlabs/build-ppc64e	warning	build succeeded but added 1 new sparse warning(s)
snowpatch_ozlabs/build-ppc64be	success	build succeded & removed 0 sparse warning(s)
snowpatch_ozlabs/build-ppc64le	success	build succeded & removed 0 sparse warning(s)
snowpatch_ozlabs/apply_patch	success	next/apply_patch Successfully applied

Commit Message

Christoph Hellwig

```
Powerpc has somewhat odd usage where ZONE_DMA is used for all memory on
common 64-bit configfs, and ZONE_DMA32 is used for 31-bit schemes.
```

So why Patchwork?

- Replying to every single patch by email is too much
- It's already used in the kernel
- It doesn't change the workflow of developers
  - ...unless they wanna go look at their tests
- Test results can be *submitted* from *anywhere!*
  - matches the decentralised nature of kernel development
  - just needs maintainer trust/approval

How do we make it happen?

- Get the patch (and its dependencies) from the Patchwork API
- Apply them on top of (at least) one git branch
  - If it doesn't apply, abort and push results
- Push that branch to a remote for testing
- Trigger tests on the remote
- Push results when they're done

snowpatch

- Rust program to automate patch-by-patch CI with Patchwork
- Initially written at LCA Geelong 2016
- Maintained by myself and Andrew Donnellan
- Running “in production” on linuxppc-dev & others
- Still very much in development
- GPL

Test runners

- In theory we could support anything, but…

  right now it's just Jenkins. Sorry.
- Jenkins has an API, does everything we need it to do
- Patches welcome!

Example Jenkins job: linux-sparse

Build #3438

Parameters

GIT_REF_PATCHED	kbuild_mark_prepare0_as_PHONY_to_fix_external_module_build_mbox_next
GIT_REF_BASE	next
GIT_REPO	git://github.com/ruscur/linux.git
DEFCONFIG_TO_USE	pmac32_defconfig

So what do you need?

- A local repository of whatever you're testing (Linux)
- (*optional*) Patchwork creds if you're pushing results
  - Needs project permissions (aka maintainer approval)
- Jenkins server and credentials
- A git remote that Jenkins can see

snowpatch config: basics

```
[patchwork]
url = "https://patchwork.ozlabs.org"
port = 443 # optional
token = "TOTALLY_A_REAL_TOKEN" # optional, needed for pushing results
polling_interval = 10 # polling interval in minutes

[jenkins]
url = "https://openpower.xyz"
port = 443
username = "ruscur"
token = "COMPLETELY_LEGIT_API_TOKEN"
```

snowpatch config: projects

```
[projects]
    # the name of the project must be as is in patchwork
    [projects.linuxppc-dev]
    repository = "~/linux"
    branches = [“master”, "next"]
    # test_all_branches defaults to true
    remote_name = "github"
    remote_uri = "git://github.com/ruscur/linux.git"
    push_results = true
```

snowpatch config: jobs

```
[[projects.linuxppc-dev.jobs]]
title = "build-ppc64le"
job = "snowpatch/job/snowpatch-linux-sparse"
remote = "GIT_REPO"
branch = "GIT_REF_PATCHED"
hefty = true # only run on full series if true
...
DEFCONFIG_TO_USE = "pseries_le_defconfig"
GIT_REF_BASE = "next"
```

Then you run it, and...

[15/15] arch: add pkey and rseq syscall numbers everywhere

Message ID	20190110162435.309262-16-arnd@arndb.de
State	New
Headers	show
Series	arch: synchronize syscall tables in preparation for y2038
Related	show

Checks

Context	Check	Description
snowpatch_ozlabs/checkpatch	success	total: 0 errors, 0 warnings, 0 checks, 100 lines checked
snowpatch_ozlabs/build-pmac32	success	build succeeded & removed 0 sparse warning(s)
snowpatch_ozlabs/build-ppc64e	warning	build succeeded but added 1 new sparse warning(s)
snowpatch_ozlabs/build-ppc64be	warning	build succeeded but added 1 new sparse warning(s)
snowpatch_ozlabs/build-ppc64le	warning	build succeeded but added 1 new sparse warning(s)
snowpatch_ozlabs/apply_patch	success	next/apply_patch Successfully applied

the links take you straight to the result

```
+ipc/shm.c:1335:6: warning: symbol 'compat_ksys_shmctl'
was not declared. Should it be static?
```

So why do I think this is the way to go?

- It's existing infrastructure (kernel.org & ozlabs.org)
- As long as the maintainer approves, results can come from anywhere
- Whatever you want to do, snowpatch makes it easy to do it on a patch-by-patch (or series-by-series) basis
- It compliments mailing lists, without replacing them or spamming them

What else can you do with this?

- Well, you don't *have* to make results public...
  - Use a public Patchwork and don't report
  - Run your own internal Patchwork to easily find results
- Find performance regressions!
  - maybe even for your userspace program
- Do whatever you want!

Thanks for listening!

Any questions?

github.com/ruscur/snowpatch

@russelldotcc

ruscur@russell.cc