Kernel Email Tools

New tools improve patch submission, review, and other processes

ELCE 2020
virtual

Frank Rowand, Sony

October 27, 2020
201017_1255
Motivation

Changes to the Linux kernel source are submitted in patch emails

Q. Is this process efficient and easy to use?

A. There have been complaints about ease of use :-)

Overview of patch flow

The following slide is a simplified conceptual view of

- patch create
- patch submit
- patch apply
What could possibly go wrong?

“It is a simple process.”

- Frank (me)  :)
What could possibly go wrong (1)?

But the reality is a complex manual process
- human errors
- difficult to use
- usability != good
What could possibly go wrong (2)?

What comes out may not look like what went in

The data may be transformed at various points in the flow.

The transformations may or may not be under your control.
What could possibly go wrong (2)?

Given that data transformations occur
- transformed patch can often be fixed
- manual fixup is extra work
- maintainers tend to develop their own scripts and process to solve the issue
2019 - State of git tools

Some of the tools provided by git to improve process
  git-format-patch
  git-send-email
  git-am
Hi, all:

It's now official, so you can use lore.kernel.org in your work without worrying that it'll be offline or down for more configuration. There is a corresponding news piece posted here:

https://www.kernel.org/lore.html
Hi, all:

I'd like your opinion on this quick helper script I wrote that uses any message-id to grab a full thread from lore.kernel.org and save it as a mbox file.

What's more useful, it can also prepare a mbox that you can pass straight to git-am:
Please give it a try and let me know your feedback! You can find it here: [ URL elided ]

It's a bit raw around the edges and I keep finding corner-cases, so it would be super interesting for me to get input from people actually trying to use it for managing patch series.
Setting expectations

The tool is performing a difficult task, analyzing
- human formatted emails and email subjects
- emails touched by various email clients and servers, with all the usual transformations

So expect b4 hiccups...

The tool author, Konstantin, is very responsive to reported issues, reported bugs, and suggestions
While working on patch attestation stuff, it became clear to me that it would be much easier if get-lore-mbox was an actual python project instead of a standalone script. As a bonus, such change would also make the tool easier to install, upgrade and package. So, say goodbye to get-lore-mbox and say hello to b4.
b4 appears to be a small factor

But the “small addition” has an immense impact on improving the workflow
But wait, there’s more

Note that b4 has features beyond replacing the receiving email server and email client.

These features are not explored here.
$ b4 --version
0.5.2

$ # requires python 3, version >= 3.6

$ b4 --help
usage: b4 [-h] [--version] [-d] [-q] {mbox,am,attest,attverify,pr,ty,diff} ...

A tool to work with public-inbox patches

positional arguments:
  {mbox,am,attest,attverify,pr,ty,diff}
    sub-command help
    mbox                Download a thread as an mbox file
    am                  Create an mbox file that is ready to git-am
    attest              Submit cryptographic attestation for patches
    attverify           Verify cryptographic attestation of patches in an mbox
    pr                  Fetch a pull request found in a message ID
    ty                  Generate thanks email when something gets merged/applied
    diff                Show a range-diff to previous series revision
$ b4 --help
usage: b4 [-h] [--version] [-d] [-q] {mbox,am,attest,attverify,pr,ty,diff} ...

A tool to work with public-inbox patches

positional arguments:
  {mbox,am,attest,attverify,pr,ty,diff}                sub-command help
    mbox                Download a thread as an mbox file
    am                  Create an mbox file that is ready to git-am
    attest              Submit cryptographic attestation for patches
    attverify           Verify cryptographic attestation of patches in an mbox
    pr                  Fetch a pull request found in a message ID
    ty                  Generate thanks email when something gets merged/applied
    diff                Show a range-diff to previous series revision

These sub-commands more useful for maintainers
I will focus on submitter use
b4 mbox

Acquires a patch series email thread into an mbox file
$ b4 mbox --help
        [-m LOCALMBOX] [-C]
        [msgid]

positional arguments:
    msgid                 Message ID to process, or pipe a raw message

optional arguments:
    -h, --help            show this help message and exit
    -o OUTDIR, --outdir OUTDIR
                          Output into this directory (or use - to output mailbox
                          contents to stdout)
    -p USEPROJECT, --use-project USEPROJECT
                          Use a specific project instead of guessing (linux-mm,
                          linux-hardening, etc)
    -c, --check-newer-revisions
                          Check if newer patch revisions exist
    -n WANTNAME, --mbox-name WANTNAME
                          Filename to name the mbox file
    -m LOCALMBOX, --use-local-mbox LOCALMBOX
                          Instead of grabbing a thread from lore, process this
                          mbox file
    -C, --no-cache        Do not use local cache
patch series email thread

$ mkdir tmp
$ cd tmp
$ b4 mbox 1602149181-237415-5-git-send-email-john.garry@huawei.com
Looking up https://lore.kernel.org/r/1602149181-237415-5-git-send-email-john.garry%40huawei.com
Grabbing thread from lore.kernel.org/linux-arm-kernel
6 messages in the thread
Saved ./1602149181-237415-5-git-send-email-john.garry@huawei.com.mbx
$ ls
1602149181-237415-5-git-send-email-john.garry@huawei.com.mbx
Where does msgid come from?
Original patch series email thread

A screenshot of the patch thread emails in my Thunderbird email client is shown on the following slides.

The entire patch series thread is the emails inside the red box (as shown above).

A random email within the thread is inside the blue box.
Original patch series email thread

A screenshot of the patch thread emails in my Thunderbird email client is shown on the following slides

The entire patch series thread is the emails inside the red box (as shown above)

A random email within the thread is inside the blue box
From John Garry <john.garry@huawei.com>


To Frank.li@nxp.com, will@kernel.org, mark.rutland@arm.com, rohidt@kernel.org, zhang

Cc linux-arm@lists.freebsd.org, device-tree@vger.kernel.org, linux-kernel@vger.kernel.org, linux-a

SMU_PMCG_IIDR was added in the SMMu3.3 spec.

For the perf tool to know the specific HW implementation, expose the PMCG_IIDR contents only when set.

Signed-off-by: John Garry <john.garry@huawei.com>

diff --git a/drivers/perf/arm_smmu3_pmuc.c b/drivers/perf/arm_smmu3_pmuc.c
index 5274f7f63590..74470b322c3 100644
--- a/drivers/perf/arm_smmu3_pmuc.c
+++ b/drivers/perf/arm_smmu3_pmuc.c
get msgid from email header

Thunderbird provides access to headers in various ways:

(1) View -> Headers -> All
(2) View -> Message Source

Subject: [PATCH v2 4/4] perf/smmuv3: Support sysfs identifier file
Date: Thu, 8 Oct 2020 17:26:21 +0800
Message-ID: <1602149181-237415-5-git-send-email-john.garry@huawei.com>

This msgid value will be used for b4 commands in the following slides
patch series email thread (repeat)

$ mkdir tmp
$ cd tmp
$ b4 mbox 1602149181-237415-5-git-send-email-john.garry@huawei.com
Looking up https://lore.kernel.org/r/1602149181-237415-5-git-send-email-john.garry%40huawei.com
Grabbing thread from lore.kernel.org/linux-arm-kernel
6 messages in the thread
Saved ./1602149181-237415-5-git-send-email-john.garry@huawei.com.mbx
$ ls
1602149181-237415-5-git-send-email-john.garry@huawei.com.mbx
patch series email thread

Use normal email client or text tools to examine .mbx
  - cover letter email
  - the patch emails
  - reply emails

eg: mutt, elm, head, cat, more, less, vi
patch series email thread

----- The mbox is the cover letter email, patch emails, replies -----

$ mutt -f 1602149181-237415-5-git-send-email-john.garry@huawei.com.mbx
Acquires a patch series into an mbox file and the cover letter (patch 0) into a .cover file (an mbox file missing one header line...)

The mbox file is just the patches, no reply emails. Thus can be used as input to ‘git am’
$ b4 am --help

[-P CHERRYPICK] [-g] [-3] [--no-cover]
[msgid]

positional arguments:
msgid                 Message ID to process, or pipe a raw message

optional arguments:
-h, --help            show this help message and exit
-o OUTDIR, --outdir OUTDIR
Output into this directory (or use - to output mailbox
contents to stdout)
-p USEPROJECT, --use-project USEPROJECT
Use a specific project instead of guessing (linux-mm,
linux-hardening, etc)
-c, --check-newer-revisions
Check if newer patch revisions exist
-n WANTNAME, --mbox-name WANTNAME
Filename to name the mbox file
-m LOCALMBOX, --use-local-mbox LOCALMBOX
Instead of grabbing a thread from lore, process this
mbox file
-C, --no-cache       Do not use local cache
-v WANTVER, --use-version WANTVER
          Get a specific version of the patch/series
-t, --apply-cover-trailers
          Apply trailers sent to the cover letter to all patches
-S, --sloppy-trailers
          Apply trailers without email address match checking
-T, --no-add-trailers
          Do not add or sort any trailers
-s, --add-my-sob
          Add your own signed-off-by to every patch
-l, --add-link
          Add a lore.kernel.org/r/ link to every patch
-Q, --quilt-ready
          Save mbox patches in a quilt-ready folder
-P CHERRYPICK, --cherry-pick CHERRYPICK
          Cherry-pick a subset of patches (e.g. "-P 1-2,4,6-",
          
          
          "-P _" to use just the msgid specified, or "-P
          *globbing*" to match on commit subject)
-g, --guess-base
          Try to guess the base of the series (if not specified)
-3, --prep-3way
          Prepare for a 3-way merge (tries to ensure that all
          index blobs exist by making a fake commit range)
--no-cover
          Do not save the cover letter (on by default when using
          -o -)
Look at patch series

$ mkdir tmp
$ cd tmp
$ b4 am 1602149181-237415-5-git-send-email-john.garry@huawei.com
Looking up https://lore.kernel.org/r/1602149181-237415-5-git-send-email-john.garry%40huawei.com
Grabbing thread from lore.kernel.org/linux-arm-kernel
Analyzing 6 messages in the thread
---
Writing ./v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.mbx
  [PATCH v2 1/4] drivers/perf: hisi: Add identifier sysfs file
  [PATCH v2 2/4] bindings/perf/imx-ddr: update compatible string
---
Total patches: 4
---
Cover: ./v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.cover
Link: https://lore.kernel.org/r/1602149181-237415-1-git-send-email-john.garry@huawei.com
Base: not found
$ git am ./v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.mbx
$ ls
v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.cover
v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.mbx
Look at patch series

$ b4 am 1602149181-237415-5-git-send-email-john.garry@huawei.com

-----  This mbox file is just the patch emails, no replies  -----  

Writing ./v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.mbx
  [PATCH v2 1/4] drivers/perf: hisi: Add identifier sysfs file
  [PATCH v2 2/4] bindings/perf/imx-ddr: update compatible string

-----  This text file is just the cover letter  -----  

Cover: ./v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.cover
Look at patch series

Use email client and / or normal text tools to examine
  - cover letter email
  - the patch emails, no replies

eg: head, cat, more, less, vi, mutt
Look at patch series

-----  This text file is just the cover letter email  -----
Look at patch series

-----  This file is just the patch emails, no replies  -----  

$ mutt -f v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.mbx
Repeat - created mbox file

$ mkdir tmp
$ cd tmp
$ b4 am 1602149181-237415-5-git-send-email-john.garry@huawei.com
Looking up https://lore.kernel.org/r/1602149181-237415-5-git-send-email-john.garry%40huawei.com
Grabbing thread from lore.kernel.org/linux-arm-kernel
Analyzing 6 messages in the thread
---
Writing ./v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.mbx
  [PATCH v2 1/4] drivers/perf: hisi: Add identifier sysfs file
  [PATCH v2 2/4] bindings/perf/imx-ddr: update compatible string
---
Total patches: 4
---
Cover: ./v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.cover
  Link: https://lore.kernel.org/r/1602149181-237415-1-git-send-email-john.garry@huawei.com
  Base: not found
    git am ./v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.mbx
$ ls
v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.cover
v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.mbx
Apply patch series mbox file (1/2)

$ b4 am 1602149181-237415-5-git-send-email-john.garry@huawei.com
   git am ./v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.mbx

$ git checkout -b sysfs_id_v2
Switched to a new branch 'sysfs_id_v2'
$ vi Makefile
$ git add Makefile
$ git commit -m "hack kernel version"
[sysfs_id_v2 0de2bfc77c92] hack kernel version
  1 file changed, 1 insertion(+), 1 deletion(-)
$ git tag sysfs_id_v2_before

$ git am ./v2_20201008_john_garry_perf_drivers_add_sysfs_identifier_file.mbx
Applying: drivers/perf: hisi: Add identifier sysfs file
Applying: bindings/perf/imx-ddr: update compatible string
Applying: perf/imx_ddr: Add system PMU identifier for userspace
Applying: perf/smmuv3: Support sysfs identifier file
Apply patch series mbox file (2/2)

$ git log --oneline sysfs_id_v2_before..
902b133763c4 (HEAD -> sysfs_id_v2) perf/smmuv3: Support sysfs identifier file
09c975368172 perf/imx_ddr: Add system PMU identifier for userspace
d5c1cfbce93b bindings/perf/imx-ddr: update compatible string
124f112bf63b drivers/perf: hisi: Add identifier sysfs file
Apply patch series mbox file (1/3)

-----  apply in a single step (shortcut)  -----  

$ git checkout -b sysfs_id_v2
Switched to a new branch 'sysfs_id_v2'
$ vi Makefile
$ git add Makefile
$ git commit -m "hack kernel version"
[sysfs_id_v2 a183c0751842] hack kernel version
  1 file changed, 1 insertion(+), 1 deletion(-)
$
$ git tag sysfs_id_v2_before
Apply patch series mbox file (2/3)

----- apply in a single step (shortcut) -----

$ b4 -q am -o - 1602149181-237415-5-git-send-email-john.garry@huawei.com | git am
Grabbing thread from lore.kernel.org/linux-arm-kernel
Writing /tmp/tmpktxc3i4b4-am-stdout
Total patches: 4
---
  Link: https://lore.kernel.org/r/1602149181-237415-1-git-send-email-john.garry@huawei.com
  Base: not found (applies clean to current tree)
Applying: drivers/perf: hisi: Add identifier sysfs file
Applying: bindings/perf/imx-ddr: update compatible string
Applying: perf/imx_ddr: Add system PMU identifier for userspace
Applying: perf/smmuv3: Support sysfs identifier file
Apply patch series mbox file (2/3)

-----  apply in a single step (shortcut)  -----  

The magic step, pulled out from the noise on the previous slide:

```bash
$ b4 -q am -o - 1602149181-237415-5-git-send-email-john.garry@huawei.com | git am
```
Apply patch series mbox file (3/3)

-----  apply in a single step (shortcut)  -----  

Applying: drivers/perf: hisi: Add identifier sysfs file
Applying: bindings/perf/imx-ddr: update compatible string
Applying: perf/imx_ddr: Add system PMU identifier for userspace
Applying: perf/smmuv3: Support sysfs identifier file

$ git log --oneline sysfs_id_v2_before..
ccfc5f946ad9 (HEAD -> sysfs_id_v2) perf/smmuv3: Support sysfs identifier file
5383f73bfd83 perf/imx_ddr: Add system PMU identifier for userspace
742bcb871280 bindings/perf/imx-ddr: update compatible string
042d4d7fb4e5 drivers/perf: hisi: Add identifier sysfs file
b4 diff

Does a ‘git range-diff’

Due to time constraints, I refer you to Konstantin’s animated example at:

https://asciinema.org/a/333888

Some more info is also provided in the resources end slides
$ b4 diff --help
usage: b4 diff [-h] [-g GITDIR] [-p USEPROJECT] [-C]
[-v WANTVERS [WANTVERS ...]] [-n] [-o OUTDIFF] [-c]
[-m AMBOX AMBOX]
msgid

positional arguments:
msgid   Message ID to process, or pipe a raw message

optional arguments:
-h, --help           show this help message and exit
-g GITDIR, --gitdir GITDIR
Operate on this git tree instead of current dir
-p USEPROJECT, --use-project USEPROJECT
Use a specific project instead of guessing (linux-mm,
linux-hardening, etc)
-C, --no-cache        Do not use local cache
-v WANTVERS [WANTVERS ...], --compare-versions WANTVERS [WANTVERS ...]
Compare specific versions instead of latest and one
before that, e.g. -v 3 5
-n, --no-diff         Do not generate a diff, just show the command to do it
-o OUTDIFF, --output-diff OUTDIFF
Save diff into this file instead of outputting to stdout
-c, --color            Force color output even when writing to file
-m AMBOX AMBOX, --compare-am-mboxes AMBOX AMBOX
                   Compare two mbx files prepared with "b4 am"
Making patches b4 friendly

Follow the patch email SUBJECT rules for [PATCH ...]

- v2, v3, ..., vn
- X/Y: 0/3, 1/3, 2/3, 3/3
- For X > 0, email is a reply to 0/Y
- Include all X/Y that still exist in the next version
How can you help improve b4?

Report issues
tools@linux.kernel.org
https://linux.kernel.org/g/tools

The tool author, Konstantin, is very responsive

Development has been quite active in 2020
b4 is your friend

- easier to acquire patch email threads
- easier to acquire patch series
- easier to apply patch series
- other more advanced features (use --help to explore the many options)
THE END

Thank you for your attention...
Questions?
How to get a copy of the slides

1) frank.rowand@sony.com

2) https://elinux.org/ELC_Presentations
   follow ‘ELC Europe 2020 Presentations’ link

3) http://events.linuxfoundation.org
Resources
List archives on lore.kernel.org

You may access the archives of many Linux development mailing lists on lore.kernel.org. Most of them include a full archive of messages going back several decades.

- listing of currently hosted archives

If you would like to suggest another kernel development mailing list to be included in this list, please follow the instructions on the following wiki page:

- Adding list archives to lore.kernel.org
lore mail list archives

List of archived mail lists
https://lore.kernel.org/lists.html

Adding list archives to lore.kernel.org
https://korg.docs.kernel.org/lore.html

More info about lore
https://www.kernel.org/lore.html
b4 install, upgrade (git repo)

download:
  git clone https://git.kernel.org/pub/scm/utils/b4/b4.git

Listing versions
  git tag -l

Example of picking a version (v0.5.2)
  git checkout -b v0.5.2 v0.5.2

Invoking b4:
  b4.sh
b4 install, upgrade (using pip)

Installing:

    python3 -m pip install --user b4

Upgrading:

    python3 -m pip install --user --upgrade b4

If ~/.local/bin is not in your PATH, you can either set a symlink to b4 there from your usual PATH, use an alias, or add ~/.local/bin to your path.
bug reports, issues

Use ‘tools’ mail list for b4 bug reports or problems
https://linux.kernel.org/g/tools
$ b4 --version
0.5.2

$ b4 --help
usage: b4 [-h] [--version] [-d] [-q] {mbox,am,attest,attverify,pr,ty,diff} ...

A tool to work with public-inbox patches

positional arguments:
  {mbox,am,attest,attverify,pr,ty,diff}

  sub-command help
  mbox            Download a thread as an mbox file
  am              Create an mbox file that is ready to git-am
  attest          Submit cryptographic attestation for patches
  attverify       Verify cryptographic attestation of patches in an mbox
  pr              Fetch a pull request found in a message ID
  ty              Generate thanks email when something gets merged/applied
  diff            Show a range-diff to previous series revision
$ b4 mbox --help
       [-m LOCALMBOX] [-C]
       [msgid]

positional arguments:
  msgid                 Message ID to process, or pipe a raw message

optional arguments:
  -h, --help            show this help message and exit
  -o OUTDIR, --outdir OUTDIR
                        Output into this directory (or use - to output mailbox
                        contents to stdout)
  -p USEPROJECT, --use-project USEPROJECT
                        Use a specific project instead of guessing (linux-mm,
                        linux-hardening, etc)
  -c, --check-newer-revisions
                        Check if newer patch revisions exist
  -n WANTNAME, --mbox-name WANTNAME
                        Filename to name the mbox file
  -m LOCALMBOX, --use-local-mbox LOCALMBOX
                        Instead of grabbing a thread from lore, process this
                        mbox file
  -C, --no-cache        Do not use local cache
b4 am --help


positional arguments:
  msgid                 Message ID to process, or pipe a raw message

optional arguments:
  -h, --help            show this help message and exit
  -o OUTDIR, --outdir OUTDIR
                         Output into this directory (or use - to output mailbox contents to stdout)
  -p USEPROJECT, --use-project USEPROJECT
                         Use a specific project instead of guessing (linux-mm, linux-hardening, etc)
  -c, --check-newer-revisions
                         Check if newer patch revisions exist
  -n WANTNAME, --mbox-name WANTNAME
                         Filename to name the mbox file
  -m LOCALMBOX, --use-local-mbox LOCALMBOX
                         Instead of grabbing a thread from lore, process this mbox file
-C, --no-cache        Do not use local cache
-v WANTVER, --use-version WANTVER
    Get a specific version of the patch/series
-t, --apply-cover-trailers
    Apply trailers sent to the cover letter to all patches
-S, --sloppy-trailers
    Apply trailers without email address match checking
-T, --no-add-trailers
    Do not add or sort any trailers
-s, --add-my-sob      Add your own signed-off-by to every patch
-l, --add-link        Add a lore.kernel.org/r/ link to every patch
-Q, --quilt-ready     Save mbox patches in a quilt-ready folder
-P CHERRYPICK, --cherry-pick CHERRYPICK
    Cherry-pick a subset of patches (e.g. "-P 1-2,4,6-", "-P _" to use just the msgid specified, or "-P *globbing*" to match on commit subject)
-g, --guess-base      Try to guess the base of the series (if not specified)
-3, --prep-3way       Prepare for a 3-way merge (tries to ensure that all index blobs exist by making a fake commit range)
--no-cover            Do not save the cover letter (on by default when using 
-o -)
$ b4 attest --help

positional arguments:
   patchfile             Patches to attest

optional arguments:
   -h, --help            show this help message and exit
   -f SENDER, --from SENDER
                         Use a custom From field
   -n, --no-submit       Do not submit attestation, just save the message ready to send
   -o OUTPUT, --output OUTPUT
                         Save attestation message in this file if not submitting it
$ b4 attverify --help

positional arguments:
  mbox                        Mbox containing patches to attest

optional arguments:
  -h, --help                 show this help message and exit
  -i ATTFILE, --attestation-file ATTFILE
                             Use this file for attestation data instead of querying lore.kernel.org
  -t, --tofu                 Force TOFU trust model (otherwise uses your global GnuPG setting)
  -X, --no-fast-exit        Do not exit after first failure
  -F, --ignore-from-mismatch
                             Ignore mismatches between From: and PGP uid data
usage: b4 pr [-h] [-g GITDIR] [-b BRANCH] [-c] [-e] [-o OUTMBOX] [msgid]

positional arguments:
  msgid                      Message ID to process, or pipe a raw message

optional arguments:
  -h, --help                show this help message and exit
  -g GITDIR, --gitdir GITDIR  Operate on this git tree instead of current dir
  -b BRANCH, --branch BRANCH Check out FETCH_HEAD into this branch after fetching
  -c, --check               Check if pull request has already been applied
  -e, --explode             Convert a pull request into an mbox full of patches
  -o OUTMBOX, --output-mbox OUTMBOX
                              Save exploded messages into this mailbox (default: msgid.mbx)
$ b4 ty --help

optional arguments:
  -h, --help            show this help message and exit
  -g GITDIR, --gitdir GITDIR
                        Operate on this git tree instead of current dir
  -o OUTDIR, --outdir OUTDIR
                        Write thanks files into this dir (default=.)
  -l, --list            List pull requests and patch series you have retrieved
  -s SEND, --send SEND  Generate thankyouys for specific entries from -l (e.g.: 1,3-5,7-; or "all")
  -d DISCARD, --discard DISCARD
                        Discard specific messages from -l (e.g.: 1,3-5,7-; or "all")
  -a, --auto            Use the Auto-Thankanator to figure out what got applied/merged
  -b BRANCH, --branch BRANCH
                        The branch to check against, instead of current
  --since SINCE         The --since option to use when auto-matching patches (default=1.week)
$ b4 diff --help
usage: b4 diff [-h] [-g GITDIR] [-p USEPROJECT] [-C]
 [-v WANTVERS [WANTVERS ...]] [-n] [-o OUTDIFF] [-c]
 [-m AMBOX AMBOX]
 [msgid]
positional arguments:
  msgid                 Message ID to process, or pipe a raw message
optional arguments:
  -h, --help            show this help message and exit
  -g GITDIR, --gitdir GITDIR
          Operate on this git tree instead of current dir
  -p USEPROJECT, --use-project USEPROJECT
          Use a specific project instead of guessing (linux-mm,
                      linux-hardening, etc)
  -C, --no-cache        Do not use local cache
  -v WANTVERS [WANTVERS ...], --compare-versions WANTVERS [WANTVERS ...]
          Compare specific versions instead of latest and one
          before that, e.g. -v 3 5
  -n, --no-diff         Do not generate a diff, just show the command to do it
  -o OUTDIFF, --output-diff OUTDIFF
          Save diff into this file instead of outputting to
                      stdout
-c, --color   Force color output even when writing to file
-m AMBOX AMBOX, --compare-am-mboxes AMBOX AMBOX
   Compare two mbx files prepared with "b4 am"
b4 diff - additional info (1 of 2)

Date: Mon, 25 May 2020 16:33:41 -0400
From: "Konstantin Ryabitsev" <konstantin@linuxfoundation.org>
To: users@linux.kernel.org, tools@linux.kernel.org
Subject: [kernel.org users] b4 version 0.5.0 is available
Message-ID: <20200525203341.z4dollmzbo7exnhc@chatter.i7.local>

b4 diff:

allows to display range-diffs between patch/series revisions. It accomplishes this by creating two "fake" detached ranges consisting solely of the files in the patches and then running a "git range-diff" between those ranges. In order to work, the following conditions must all be met:

1. patches must contain "index" blob information
2. we must be able to find all of those blobs in the repo
Behind the scenes, b4 will create a temporary worktree containing just the files mentioned in the patch and run "git am" to create the commit range. After deleting the worktree, this range will continue to exist in the repo as loose objects (until gc cleans them up), allowing to run git range-diff across both patch series.

In my experience, this works about 60% of the time, but I expect this percentage will be higher for you, since you are more likely to have the needed blobs in your repo due to using the same tree as the submitter.
Corrigendum

The following slides are the changes to the slides done after the ELCE 2020 video presentation was recorded

Format:
( old slide number / new slide number ) change
(30 / 30-34) expand single slide to multiple slides to better explain how to get msgid

(31 / 35) add 'repeat' to title

(32 - 33 / 36-37) reverse order of the two slides