preparing and sending patches

Abdul Rahim

23 July 2024

preparing patches

- ensure that your code conforms **linux kernel coding style**
- a script `checkpatch.pl` is provided; which you can run off of `diff`

```
git diff /path/to/changed/file  /tmp/df; \
    scripts/checkpatch.pl /tmp/df
```

- run checkpatch.pl **before testing or commmiting** changes
- make necessary changes and fix warning/errors generated by the script

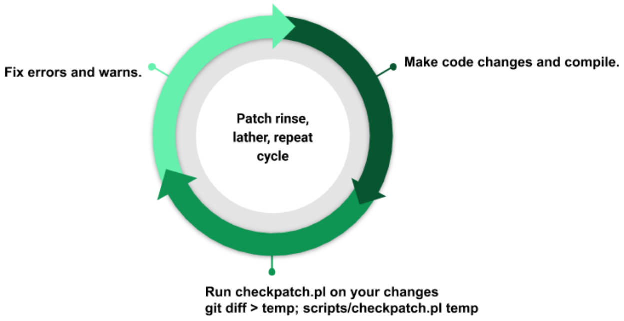
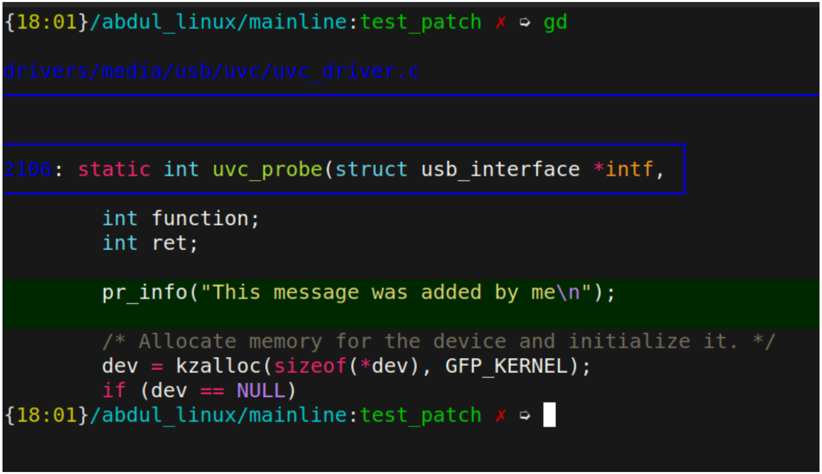

Patch rinse, lather, repeat cycle
Make code changes and compile.
Run checkpatch.pl on your changes
git diff > temp; scripts/checkpatch.pl temp
Fix errors and warns.

Figure 1: patch workflow

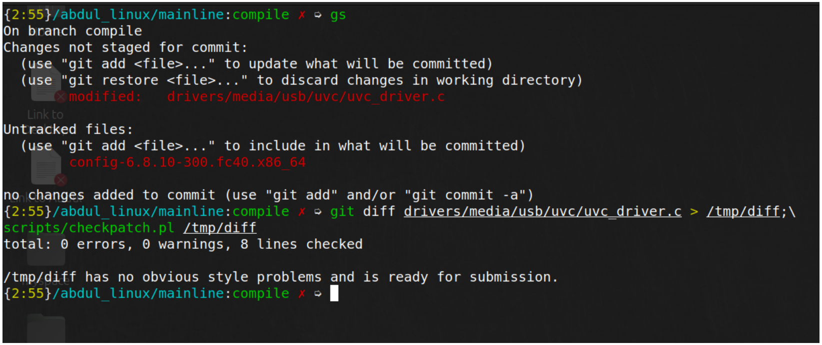
once checkpatch is happy; test your changes and commit your change

```
git commit -a # to commit all changes
git commit <filenames> # in case you want to
    send changes as seprate patches
```



```
{18:01}/abdul_linux/mainline:test_patch ✗ ➪ gd

drivers/media/usb/uvc/uvc_driver.c


2106: static int uvc_probe(struct usb_interface *intf,

        int function;
        int ret;

        pr_info("This message was added by me\n");

        /* Allocate memory for the device and initialize it. */
        dev = kzalloc(sizeof(*dev), GFP_KERNEL);
        if (dev == NULL)
{18:01}/abdul_linux/mainline:test_patch ✗ ➪
```

Figure 2: diff output



```
{2:55}/abdul_linux/mainline:compile ✗ ➭ gs
On branch compile
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   drivers/media/usb/uvc/uvc_driver.c

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        config-6.8.10-300.fc40.x86_64

no changes added to commit (use "git add" and/or "git commit -a")
{2:55}/abdul_linux/mainline:compile ✗ ➭ git diff drivers/media/usb/uvc/uvc_driver.c > /tmp/diff;\
scripts/checkpatch.pl /tmp/diff
total: 0 errors, 0 warnings, 8 lines checked

/tmp/diff has no obvious style problems and is ready for submission.
{2:55}/abdul_linux/mainline:compile ✗ ➭
```

Figure 3: checkpatch script gives no warning

commit conventions

- commit message has a short subject; and a description/longer commit message;
- Explain: **why the code change is needed**, instead of **what code does**. Checkout *how to write a git commit*
- begin the commit message with a single short (less than 50 character) line summarizing the change, followed by a blank line and then a more thorough description.
- The **text up to the first blank line** in a commit message is treated as the commit **title**, and that title is used throughout Git. For example, **Git-format-patch(1) turns a commit into email, and it uses the title on the Subject line and the rest of the commit in the body.**

generate patch

- After committing the change, generate the patch running the following command:

```
git format-patch -1 <commit ID>
```

- do not use the dont use –pretty option. Because then `git send-email` doesn't work.

sending your patch

send your patch using the command:

```
git send-email --to=email@gmail.com /path/to/
    patch
```