Contributing to Linux Kernel

Vaishali Thakkar
(https://vthakkar1994.wordpress.com/, @kernel_girl)
Tapasweni Pathak
(https://tapasweni-pathak.github.io/)

https://github.com/vthakkar1994/Linux-Kernel-Workshop

What you need

- Git
- Source-code of linux-next/staging-testing
- Your favorite text-editor [Vim]
- Mail transport client esmtp
- Mail client mutt or git send-email for sending patches
- Instructions available at:

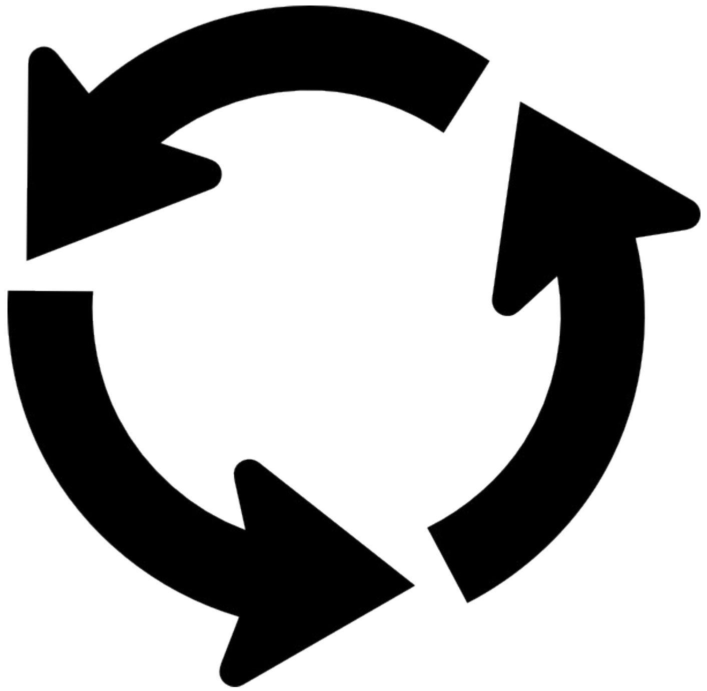
https://github.com/vthakkar1994/Linux-Kernel-Workshop

What we are going to cover

- Work flow cycle of the Linux kernel
- Linux kernel communication and coding style
- Git basics
- Understanding patch best practices with generating and mailing first patch
- What after sending first patch?

The process of the kernel hacking is a

CYCLE



The creative cycle

- Code your changes

The creative cycle

- Code your changes
  - Find a contribution to make

The creative cycle

- Code your changes
  - Find a contribution to make
  - Read mailing list archives

The creative cycle

- Code your changes
  - Find a contribution to make
  - Read mailing list archives
  - Gain experience and ask questions

The creative cycle

- Code your changes
- Create and send in your patch

The creative cycle

- Code your changes
- Send in your patch
- Gather feedback
    - Testing results and the patch

The creative cycle

- Code your changes
- Send in your patch
- Gather feedback
  - Testing results and the patch
  - Mentoring and guidance

The creative cycle

- Code your changes
- Send in your patch
- Gather feedback
  - Testing results and the patch
  - Mentoring and guidance
  - Discussion of strategies and suggestions

The creative cycle

- Code your changes
- Send in your patch
- Gather feedback
- Repeat

Communication style

- Mailing lists – scripts/get_maintainer.pl

Communication style

- Mailing lists – scripts/get_maintainer.pl
- Responding to emails
  - Consistent communictaion style

    - Use factual, objective language
    - Be considerate and polite
    - People are busy, just be patient

Communication style

- Mailing lists – scripts/get_maintainer.pl
- Responding to emails
  - ✔ Consistent communication style
  - Respond inline – Say NO to top-posting

```
From: Kludge Crufty <example@email.com>
Subject: Design decisions for next release
On Fri, Sep 12, 2014 at 03:00:56PM -0700, Baz Quux wrote:
> On Fri, 12 September 2014 at 02:30:17PM -0700, Foo Bar wrote:
>
>  I think we should do X.

 I think we should do Y.
 Kludge
```

Communication style

- Mailing lists – scripts/get_maintainer.pl
- Responding to emails
  - ✔ Consistent communication style
  - ✔ Respond inline
  - Make sure to use one of the standard email clients listed in Documentation/email-clients.txt.

    1. Mutt

       - sudo apt-get install mutt

    2. git send-email

       - sudo apt-get install git-email

    3. Thunder bird

Communication style

- Mailing lists – scripts/get_maintainer.pl
- Responding to emails
  - ✔ Consistent communication style
  - ✔ Respond inline
  - ✔ Use standard email clients

    - Do NOT use the gmail web interface

      WHY: It line-wraps the mail

Communication style

- Mailing lists – scripts/get_maintainer.pl
- Responding to emails
  - ✔ Consistent communication style
  - ✔ Respond inline
  - ✔ Use standard email clients
  - ✗ Do NOT use the gmail web interface
  - - DO NOT use outlook

    WHY: It mangles patches (turns tabs into spaces).

Communication style

- Mailing lists – scripts/get_maintainer.pl
- Responding to emails
  - ✔ Consistent communication style
  - ✔ Respond inline
  - ✔ Use standard email clients
  - ✗ Do NOT use the gmail web interface
  - ✗ DO NOT use outlook
  - ✗ Don't include quotes in your signature

Communication style

- Mailing lists – scripts/get_maintainer.pl
- Responding to emails
- Avoid sending private mails

  - Likely to be missed

  - Not encouraged by developers/maintainers

Communication style

- Mailing lists – scripts/get_maintainer.pl
- Responding to emails
- Avoid sending private mails
- Internet Relay Chat [IRC]
  - Looks like multi-way text messaging
  - Use a dedicated client [not a web client]
  - Connect to a network
  - Once on a network, join a channel

Communication style

- Mailing lists – scripts/get_maintainer.pl
- Responding to emails
- Avoid sending private mails
- IRC Ettiquetts
  - NEVER SHOUT! Using all capital letters is the same as screaming

Communication style

- Mailing lists – scripts/get_maintainer.pl
- Responding to emails
- Avoid sending private mails
- IRC Ettiquetts
  - ✗ NEVER SHOUT! Using all capital letters is the same as screaming
  - ● Be considerate. When you are asking for help, being rude or pushy will rarely get you an answer to your question.

Communication style

- Mailing lists – scripts/get_maintainer.pl
- Responding to emails
- Avoid sending private mails
- IRC Ettiquetts
  - ✗ NEVER SHOUT! Using all capital letters is the same as screaming
  - ✔ Be considerate. When you are asking for help, being rude or pushy will rarely get you an answer to your question.
  - ✔ Be patient.

Communication style

- Mailing lists – scripts/get_maintainer.pl
- Responding to emails
- Avoid sending private mails
- IRC Ettiquetts
- Remember, anything you post on the internet is there FOREVER.

Coding Style

- Documentation/codingstyle

Coding Style

- Documentation/codingstyle
- Use Tabs

Coding Style

- Documentation/codingstyle
- Use Tabs
- All tabs are 8 characters

  - 'set tabstop=8'

Coding Style

- Documentation/codingstyle
- Use Tabs
- All tabs are 8 characters
- 80 character line limit

Coding Style

- Documentation/codingstyle
- Use Tabs
- All tabs are 8 characters
- 80 character line limit
- Run scripts/checkpatch.pl [and cleanpatch] before sending any patch

git

- Open source
- Created by Linus in 2005 to work on Linux kernel
- Fastest version control system
- Installation: 'sudo apt-get install git'
- Setting up .gitconfig

Basic commands

- Git clone

  git clone git://git.kernel.org/pub/scm/linux/kernel/git/gregkh/staging.git

- Git branch

  git checkout -b first-patch [creating new branch]

- Git status

Patch

```
fcoe: Convert use of __constant_htons to htons

In little endian cases, the macro htons unfolds to __swab16 which provides special case for constants. In
big endian cases, __constant_htons and htons expand directly to the same expression. So, replace
__constant_htons with htons with the goal of getting rid of the definition of __constant_htons completely.


Signed-off-by: Vaishali Thakkar <vthakkar1994@gmail.com>

Acked-by: Vasu Dev <vasu.dev@intel.com>

Signed-off-by: Mark Brown <broonie@kernel.org>


diff --git a/drivers/scsi/fcoe/fcoe.c b/drivers/scsi/fcoe/fcoe.c

index ec193a8..d3eb80c 100644

--- a/drivers/scsi/fcoe/fcoe.c

+++ b/drivers/scsi/fcoe/fcoe.c

@@ -364,7 +364,7 @@ static int fcoe_interface_setup(struct fcoe_interface *fcoe,

         * on the ethertype for the given device

         */

        fcoe->fcoe_packet_type.func = fcoe_rcv;

-       fcoe->fcoe_packet_type.type = __constant_htons(ETH_P_FCOE);

+       fcoe->fcoe_packet_type.type = htons(ETH_P_FCOE);

        fcoe->fcoe_packet_type.dev = netdev;
```

Developer's certificate of origin

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I have the right to submit it under the open source license indicated in the file; or

(b) The contribution is based upon previous work that, to the best of my knowledge, is covered under an appropriate open source license and I have the right under that license to submit that work with modifications, whether created in whole or in part by me, under the same open source license (unless I am permitted to submit under a different license), as indicated in the file; or

(c) The contribution was provided directly to me by some other person who certified (a), (b) or and I have not modified it.

(d) I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved.

Creating and emailing first patch

- Run scripts/checkpatch.pl

  scripts/checkpatch.pl –file –terse directory/file.c

Creating and emailing first patch

- Run scripts/checkpatch.pl
- Pick one warning and change the code

Creating and emailing first patch

- Run scripts/checkpatch.pl
- Pick one warning and change the code
- Compile the change

  - make directory/file.o

Creating and emailing first patch

- Run scripts/checkpatch.pl
- Pick one warning and change the code
- Compile the change
- Create the patch
  - git add directory/file.c

Creating and emailing first patch

- Run scripts/checkpatch.pl
- Pick one warning and change the code
- Compile the change
- Create the patch
  - ✔ git add directory/file.c
  - ● git commit -s -v

    - Adds signed-off by line and shows the diff you are committing

Creating and emailing first patch

- Run scripts/checkpatch.pl
- Pick one warning and change the code
- Compile the change
- Create the patch
  - ✔ git add directory/file.c
  - ✔ git commit -s -v
  - git format-patch -o /tmp/ HEAD^

    - -o flag specifies where to put the patch

Creating and emailing first patch

- Run scripts/checkpatch.pl
- Pick one warning and change the code
- Compile the change
- Create the patch
  - ✓ git add directory/file.c
  - ✓ git commit -s -v
  - ✓ git format-patch -o /tmp/ HEAD^
  - Getting the list of maintainers

    - scripts/get_maintainer.pl /tmp/xxx.patch

Creating and emailing first patch

- Run scripts/checkpatch.pl
- Pick one warning and change the code
- Compile the change
- Create the patch
  - ✔ git add directory/file.c
  - ✔ git commit -s -v
  - ✔ git format-patch -o /tmp/ HEAD^
  - ✔ Getting the list of maintainers
  - ● Sending your patch with git send-email or mutt

    - git send-email /tmp/xxx.patch | mutt -H /tmp/xxx.patch

Creating and emailing first patch

- Run scripts/checkpatch.pl
- Pick one warning and change the code
- Compile the change
- Create the patch
  - ✓ git add directory/file.c
  - ✓ git commit -s -v
  - ✓ git format-patch -o /tmp/ HEAD^
  - ✓ Getting the list of maintainers
  - ✓ Sending your patch with mutt or git send-email

What after sending first patch?

- Repeat the work-flow cycle
- Use bug finding tools [sparse, smatch, coccinelle etc]
- Work on API functions
- Work on easy to fix issues of y2038 problem, devm functions etc
- Work on drivers or topics of your interest in the linux-kernel

Overview of common bug finding tools

- Sparse:
  - Sparse is a tool for static code analysis that helps kernel developers to detect coding errors.
  - Installation: sudo apt-get install sparse [from package manager]
  - Use: make C=2 drivers/staging/wlan-ng/
  - Dedicated mailing list - YES
  - Resources:

    - Documentation/sparse.txt

    - https://sparse.wiki.kernel.org/index.php/Main_Page

    - http://kernelnewbies.org/Sparse

Overview of common bug finding tools

- Smatch:
  - Smatch is a C static analysis tool with the lots of kernel specific checks
  - The core part of Smatch is a flow analysis engine.
  - Developed by Dan Carpenter
  - Dedicated mailing list – YES
  - Resources:

    - Oracle mainline kernel blog

    - http://smatch.sourceforge.net/

Overview of common bug finding tools

- Coccinelle:
  - Program matching and transformation tool for the C code
  - Provides the language Semantic patch language [SmPL] for specifiying desired matches and transformation
  - Initially targeted towards performing collateral evolutions
  - Beyound collateral evolutions, successfully used for bug finding and fixing in the Linux kernel
  - Dedicated mailing list and IRC channel - YES
  - Resources:

    - http://coccinelle.lip6.fr/

    - https://github.com/coccinelle

    - Conference talks by Julia Lawall and other kernel developers

    - Research papers by INRIA researchers

Overview of common bug finding tools

- Trinity
  - A Linux system call fuzz tester
  - Used by kernel developers to find the bugs
  - Developed by Dave Jones
  - Dedicated mailing list - YES
  - Resources:

    - http://codemonkey.org.uk/projects/trinity/

    - https://github.com/kernelslacker/trinity

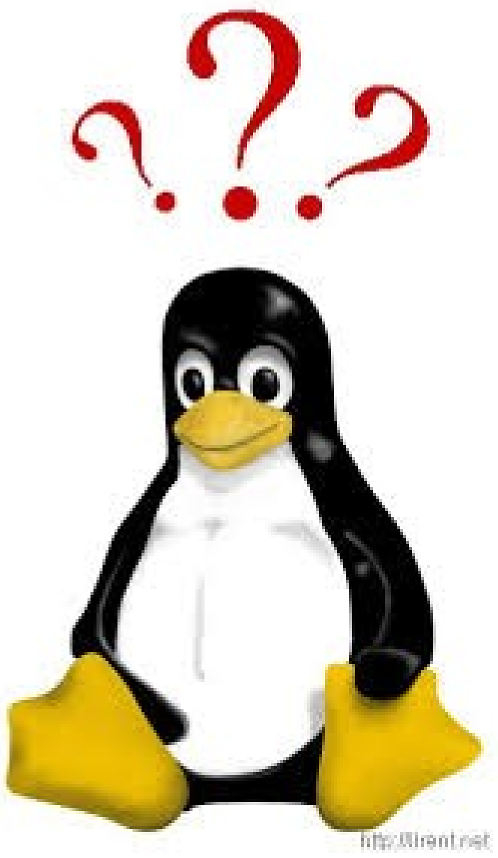
    - LWN articles

    - Conference talks by Dave Jones

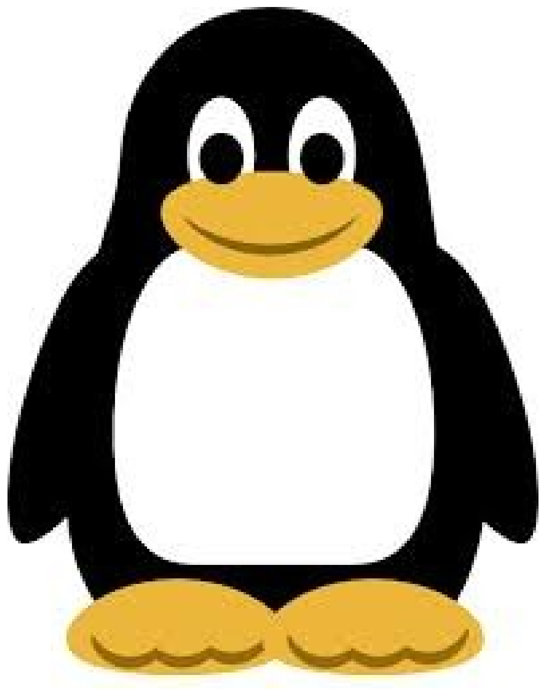
Some other bug finding tools

- Flowfinder: A simple program that examines C/C++ source code and reports possible security weaknesses (“flaws”) sorted by risk level
- Pahole: Shows and manipulates data structure layout
- Parfait: A static bug checking tool for the C and C++ code

Questions



Welcome to the Linux Kernel community



Thank You