From an Idea to a Patch in the Linux Mainline

Marta Rybczynska

#ossummit  @mrybczynska
# New kernel contributors by version

<table>
<thead>
<tr>
<th>Kernel version</th>
<th>All developers</th>
<th>First time contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.8</td>
<td>1991</td>
<td>304</td>
</tr>
<tr>
<td>5.7</td>
<td>1878</td>
<td>281</td>
</tr>
<tr>
<td>5.6</td>
<td>1712</td>
<td>214</td>
</tr>
<tr>
<td>5.5</td>
<td>1885</td>
<td>285</td>
</tr>
<tr>
<td>5.4</td>
<td>1802</td>
<td>266</td>
</tr>
<tr>
<td>5.3</td>
<td>1846</td>
<td>256</td>
</tr>
<tr>
<td>5.2</td>
<td>1716</td>
<td>245</td>
</tr>
<tr>
<td>5.1</td>
<td>1707</td>
<td>245</td>
</tr>
<tr>
<td>5.0</td>
<td>1743</td>
<td>283</td>
</tr>
</tbody>
</table>

Source: LWN.net kernel development statistics
Steps

• Pre-requirements: what do I need?
• Issue analysis: how to understand it?
• Tools: what can help me?
• Formalities: how to prepare the change?
• Procedures: how to get it accepted?
• What if it is not a bug?
Pre-requirements

What do I need?
• Bug
  – Suspicious warning
  – Something happens differently than you expect
  – BUG()
• New feature
  – New functionality in existing code
  – New subsystem or driver
• Improvement
  – Better performance (in a specific case)
  – Refactoring
  – New test
Environment

- Linux source code
- Compiler, debugger
- Test system
  - A machine with root access, a VM or an embedded system
- Text editor
  - Any text editor for developers will work, need to support raw text mode and control insertion of white spaces
  - Linux coding style function
- Email client
  - Support raw text mode
  - For sending patches
Your development system – running Linux

• Make sure you have sudo rights
  – Needed to install the kernel and modules

• Install build dependencies (differs between distros)
  – Example for Debian:

    ```
sudo apt install libncurses5-dev gcc make git exuberant-ctags
bcc libssl-dev
    ```

• Watch out for disk space
  – `/boot/` is small in many distributions – may need to resize
Linux source trees and tools

- **Linux kernel source trees**
  - Main tree <- you submit your patches there
    
    `git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git branch master`
  
  - **Stable trees**
    
  
  - Subsystem trees – depending on the subsystem you change

- **Source indexer**
  
  - Offline: your text editor, ctags
  
  - Online: [https://elixir.bootlin.com/linux/latest](https://elixir.bootlin.com/linux/latest)
Compiling the kernel – simple way

```bash
git clone
git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
cd linux
# check your current kernel
uname -r
# copy the current config
cp /boot/config-`uname -r`* .config
# use old config and add options that were not present in the old kernel with defaults
make olddefconfig
# the big moment – build it
make
# install
sudo make modules_install install
# you may need an update in the bootloader to boot it
```
Exercise: Compile and Boot a kernel

- Download the Linux master
- Change the kernel version name (EXTRAVERSION in the Makefile)
- Compile the kernel and install it
- Boot it
- Check that `uname -a` gives the new version
- Bonus points: do it on an embedded platform
Issue analysis

How to understand it?
With a BUG() example
Where to look for information?

- **Kernel documentation**
  - Documentation/ in the kernel tree
- **The code itself, and comments around**
  - Source indexer is **helpful**
- **Previous mailing list discussions**
- **News sites**
  - Subsystem description when it was introduced
  - Big interface changes...
How to read a BUG() - example from kernel 5.0

[ 300.433586] nvme nvme0: ANA log page size (8208) larger than MDTS (8192).
[ 300.435387] nvme nvme0: disabling ANA support.
[ 300.437835] nvme nvme0: creating 4 I/O queues.
[ 300.459132] nvme nvme0: new ctrl: NQN "nqn.0.0.0", addr 10.91.0.1:8009
[ 300.464609] BUG: unable to handle kernel NULL pointer dereference at 0000000000000008
[ 300.466342] #PF error: [normal kernel read fault]
[ 300.467385] PGD 0 P4D 0
[ 300.467987] Oops: 0000 [#1] SMP PTI
[...]
[ 300.471532] Workqueue: nvme-wq nvme_scan_work [nvme_core]
[ 300.472724] RIP: 0010:nvme_parse_ana_log+0x21/0x140 [nvme_core]
[...]
[ 300.477374] RSP: 0018:ffffa50e80fd7cb8 EFLAGS: 00010296
[...]
[ 300.494991] Call Trace:
[ 300.495645]  nvme_mpath_add_disk+0x5c/0xb0 [nvme_core]
[ 300.496880]  nvme_validate_ns+0x2ef/0x550 [nvme_core]
[...]
[ 300.506280] Modules linked in: nvme_tcp nvme_rdma rdma_cm [...]

Complete call trace in commit 66b20ac0a1a10769d059d6903202f53494e3d902
How to read a BUG() - example from kernel 5.0

[ 300.433586] nvme nvme0: ANA log page size (8208) larger than MDTS (8192).
[ 300.435387] nvme nvme0: disabling ANA support.
[ 300.437835] nvme nvme0: creating 4 I/O queues.
[ 300.459132] nvme nvme0: new ctrl: NQN "nqn.0.0.0", addr 10.91.0.1:8009
[ 300.464609] BUG: unable to handle kernel **NULL pointer dereference at 0000000000000008**
[ 300.466342] #PF error: [normal kernel read fault]
[ 300.467385] PGD 0 P4D 0
[ 300.467987] Oops: 0000 [#1] SMP PTI
[...]
[ 300.471532] Workqueue: nvme-wq nvme_scan_work [nvme_core]
[ 300.472724] RIP: 0010: nvme_parse_ana_log+0x21/0x140 [nvme_core]
[...]
[ 300.477374] RSP: 0018: ffffffff80fd7cb8 EFLAGS: 00010296
[...]
[ 300.494991] Call Trace:
[ 300.495645] nvme_mpath_add_disk+0x5c/0xb0 [nvme_core]
[ 300.496880] nvme_validate_ns+0x2ef/0x550 [nvme_core]
[...]
[ 300.506280] Modules linked in: nvme_tcp nvme_rdma rdma_cm [,,,]

Complete call trace in commit 66b20ac0a1a10769d059d6903202f53494e3d902

Crash in this function

And called from this one
static int nvme_parse_ana_log(struct nvme_ctrl *ctrl, void *data,
    int (*cb)(struct nvme_ctrl *ctrl, struct nvme_ana_group_desc *,
    void *))
{
    void *base = ctrl->ana_log_buf;
    size_t offset = sizeof(struct nvme_ana_rsp_hdr);
    int error, i;
    lockdep_assert_held(&ctrl->ana_lock);
    for (i = 0; i < le16_to_cpu(ctrl->ana_log_buf->ngrp); i++) {
        struct nvme_ana_group_desc *desc = base + offset;
        u32 nr_nsid = le32_to_cpu(desc->nnnsids);
        size_t nsid_buf_size = nr_nsid * sizeof(_.le32);
        if (WARN_ON_ONCE(desc->grpid == 0))
            Return -EINVAL;
        if (WARN_ON_ONCE(le32_to_cpu(desc->grpid) > ctrl->anagrpmax))
            Return -EINVAL;
        if (WARN_ON_ONCE(desc->state == 0))
            Return -EINVAL;
        if (WARN_ON_ONCE(desc->state > NVME_ANA_CHANGE))
            Return -EINVAL;
        offset += sizeof(*desc);
        if (WARN_ON_ONCE(offset > ctrl->ana_log_size - nsid_buf_size))
            Return -EINVAL;
        error = cb(ctrl, desc, data);
        if (error)
            return error;
    }

Screenshot from: https://elixir.bootlin.com/linux/v5.0.21/source/drivers/nvme/host/multipath.c#L321

Addr2line will point here
Why it isn’t allocated?

• Path 1: Check all allocation paths
• Path 2: Trace the call stack
• Path 3: Look into the kernel messages around
Source code analysis: finding the core issue

```c
int nvme_mpath_init(struct nvme_ctrl *ctrl, struct nvme_id Ctrl *id)
{
    int error;

    if (!nvme_ctrl_use_ana(ctrl))
        return 0;

    ctrl->anacap = id->anacap;
    ctrl->anatt = id->anatt;
    ctrl->nanagrpdi = le32_to_cpu(id->nanagrpdi);
    ctrl->nanagrpmax = le32_to_cpu(id->nanagrpmax);

    mutex_init(&ctrl->ana_lock);
    timer_setup(&ctrl->anatt_timer, nvme anatt timeout, 0);
    ctrl->ana_log_size = sizeof(struct nvme_ana rsp hdr) +
                          ctrl->nanagrpdi * sizeof(struct nvme ana group desc);
    ctrl->ana_log_size += ctrl->max_namespaces * sizeof(__le32);

    if (ctrl->ana_log_size > ctrl->max_hw_sectors << SECTOR_SHIFT) {
        dev_err(ctrl->device,
                "ANA log page size (%zd) larger than MDTS (%d).\n", ctrl->ana_log size,
                ctrl->max_hw_sectors < SECTOR_SHIFT);
        dev_err(ctrl->device, "disabling ANA support.\n");
        return 0;
    }

    INIT_WORK(&ctrl->ana work, nvme ana work);
    ctrl->ana_log_buf = kmalloc(ctrl->ana_log_size, GFP_KERNEL);
    if (!ctrl->ana_log_buf) {
        error = -ENOMEM;
        goto out;
    }
}
```

Messages seen in the backtrace
Early return
Allocation is later

Screenshot from: https://elixir.bootlin.com/linux/v5.0.21/source/drivers/nvme/host/multipath.c#L529
Exercise: Your Analysis of a Kernel Issue

- Find a commit fixing a bug in the subsystem you’d like to modify; Ideally with a warning or a BUG().
- Look at the problem description ONLY
- With the information in the BUG() dump try to understand what the problem may be (list a couple of ideas)
- Verify with the patch content
- Bonus points: analyze a bug that is not fixed yet
Tools

What can help me?
printf() debugging

• Actually… no printf() in the kernel
printk() debugging

- **Simple printk() - rare**
  
  - \texttt{printk(KERN_ERR "ah.. something went wrong, code: \%d
\", ret);}

- **pr_ functions: pr_err(), pr_info(), pr_debug()**
  
  - Equivalent of \texttt{printk(KERN...)}
  
  - Except for \texttt{pr_debug()}, which can be compiled out

- **Device debugging: dev_err(), dev_info(),...**
  
  - Used in device drivers, shows additional device information
  
  - Example: \texttt{dev_info(dev, "device up\n");}

- **Dynamic debug**
  
  - Enable \texttt{pr_debug()} in a specific file/place
  
  - Example: \texttt{echo -n 'file myfile +p' > \}/sys/kernel/debug/dynamic_debug/control}
Useful debugging tools

- Oops, BUG(), WARN_ON()
- Kgdb – debugger at the kernel level
- Ftrace – function tracer, allows to find out what happened between two events
- Perf – for all kind of performance measurements and counters
Testing frameworks

- Kernel selftest framework
  - Testing kernel from the user space
  - Tests run after boot
  - Can use test modules

- KUnit
  - Unit tests inside the kernel, eg. in a driver
  - Similar to typical unit testing frameworks

- Development tools documentation:
Formalities

How to prepare the change?
Linux Coding Style – Simplified to one slide

• Tabs are 8 characters
• One statement by line
• 80 characters line limit preferred maximum length
• Short names, lower case
• Braces placement like in:

```c
if (is_condition()) {
    do_something();
    and_more();
}
```
Linux Coding Style - resources

- Complete definition:
  Documentation/process/coding-style.rst
- checkpatch.pl – the tool to verify the coding style
  - For patch files: ./scripts/checkpatch.pl mypatch.patch
  - For source files: ./scripts/checkpatch.pl -f myfile.c
Formatting a patch – how?

• From a git commits with `git format-patch`
  – Example: patch from the last commit:
    `git format-patch -1`
Commit message template

subsystem: title
<empty line>
Describe what is the purpose of the patch in lines of 75 characters max
<empty line>
Signed-off-by: <developer@example.org>
somedriver: fix timer overflow after 32 minutes

This patch fixes a crash happening when a cat sleeps on the keyboard for more than 32 minutes.

Signed-off-by: <developer@example.org>
Certificate of origin

Signed-off-by: Firstname Lastname <developer@example.org>

- Serious, legal matters
- Use only real names
- Certifies that you have a right to submit under an open source license
Other frequent tags

- **Acked-by:** the person has reviewed the patch (often by maintainers)
- **Reviewed-by:** the person formally reviewed the code, they think it is ready to be included; all comments communicated to the author
- **Reported-by:** the person who found the issue
- **Tested-by:** the person tested the patch
- **Fixes:** states the original commit this one fixes

If you want to know more: Documentation/process/submitting-patches.rst
Exercise – prepare a patch

• Perform a change in the kernel
• Test it
• Use checkpatch.pl to verify if it is correct (formally)
• Format a patch file
• We can discuss it during the conference
Procedures

How to get it accepted?
Submitting a patch – where?

- ./scripts/get-maintainer.pl
Submitting a patch – where?

```
$ ./scripts/get_maintainer.pl -f ./lib/random32.c

"David S. Miller" <davem@davemloft.net> (maintainer:NETWORKING [GENERAL])
Jakub Kicinski <kuba@kernel.org> (maintainer:NETWORKING [GENERAL])
netdev@vger.kernel.org (open list:NETWORKING [GENERAL])
linux-kernel@vger.kernel.org (open list)
Submitting a patch - how? (1/2)

• Make sure the coding style is fine
• Send plain text email, inline the patch
• **Subject:** [PATCH] subsystem: title
  – The first line of your patch file
• Can use most email clients
  – With specific configuration!
  – Howto: Documentation/process/email-clients.rst
Submitting a patch - how? (2/2)

- **Possible tool to use:** git send-email
  - Especially when sending patch sets

- **NOT to do**
  - No attachments
  - No encryption, compression, legal, long signatures
  - Not github pull requests

- **Hint:** when preparing a new environment send a patch to yourself
Review process (1/2)

- Patches are rarely accepted in the first version
- Include and answer to « why? » in the patch description
- Count one week to receive comments
Review process (2/2)

- Reviewers submit comments with quotations
  - Often brief – they are busy
- Answer politely
  - You can disagree with the review
  - Use facts
  - Address the problem
- Ask for clarifications
  - « Did you mean something like this » with a code snippet
- Submit [PATCH v2] after addressing the comments
What kind of feedback can I get?

- Coding style change
- Request to refactor existing code
- Request to use existing API
- Suggestions on how to improve
- Alternative solution draft
- Request for clarification (« Why did you? »)
- Explanation why/when your solution won’t work
What if I do not get any feedback?

• Resubmit after a week
• Verify
  – Is the patch title clear?
  – Is the description clear? Does it say why the change is important?
  – Is it send to the right maintainer?
  – Is the change small enough? Should you consider a patch set?
What if it is not a bug?
Nearly the same!
New feature

- Communicate early
- Show you know the rules
  - Coding style!
  - Tests and testers
- [RFC] patch
  - RFC is Request for comments
  - Low maturity patch, does not need to be complete
  - Asking for discussion
  - Often only to the subsystem mailing list, not the whole LKML
- Split changes into logical steps – easier to review!
Splitting big changes

- **Patch set**: a set of patches submitted together
- **One logical change in a patch**
- **Separate title and description for each patch**
- **The kernel should compile and work after each patch from the set**
Splitting big changes - example 1

- Fix a bug
- Fix another bug in the same file
- Fix a comment in another file
- Add a test
Splitting big changes - example 2

- Add a new generic function with documentation
- Add another generic function in another subsystem
- Refactoring in a driver
- Use your new functions in the driver
Wrapping up...
Resources

- Kernelnewbies.org – resources for kernel developers, HOWTOS, descriptions
  - [https://kernelnewbies.org/FirstKernelPatch](https://kernelnewbies.org/FirstKernelPatch)
  - [https://kernelnewbies.org/PatchPhilosophy](https://kernelnewbies.org/PatchPhilosophy)

- [http://eudyptula-challenge.org/](http://eudyptula-challenge.org/) - exercises to get into the kernel development step by step. Can’t subscribe anymore, but you can search for the exercises

- LWN.net kernel articles – for information about the changes and how the process works

- Linux kernel mailing list archives
Summary

• New developers see their patches in each kernel release

• Start simple
  – Test your setup
  – Learn the rules of your subsystem

• First patch doesn’t have to be perfect
  – Show that you did your homework
  – Be respectful

• We learn by doing :)
Time for your questions!

From an Idea to a Patch in the Linux Mainline

Marta Rybczynska
@mrybczynska
marta@rybczynska.net