

# How to Participate in the Linux Kernel Development (and Why)

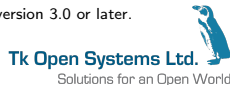
Baruch Siach  
baruch@tkos.co.il

Tk Open Systems

October 10, 2011



This work is released under the Creative Commons BY-SA version 3.0 or later.



# Introduction: The Linux Kernel Software Project

- New release every three months
- More than 8,000,000 lines of code
- About 10,000 patches in each release
- More than 1000 developers participate in each release
- You can participate too

# Why Bother?

The kernel development process tends to be time consuming and frustrating at times, so why bother? Here are a few reasons:

- Improve code quality; get code review from the experts
- Avoid common coding pitfalls:
  - Incorrect use of kernel API
  - Wrong hardware initialization sequence
- Learn better ways to do what you want
- Influence the kernel development decision making
  - The case of Linux Security Module framework (non) removal
  - Kernel people appreciate code rather than talk <sup>1</sup>
- Automatic availability of your feature to all users
- Automatic maintenance of your code
  - Internal kernel API tend to change as needed
  - Only in-kernel code gets updated when the API changes
- Bug fixes from users and developers

---

<sup>1</sup> "Talk is cheap. Show me the code." - Linus Torvalds (Aug 25, 2000)

# The Kernel Release Schedule

- ① Kernel subsystem maintainers collect reviewed and tested patches for kernel version  $N$  until the release of  $N-1$ .
- ② Linus releases Kernel version  $N-1$ ; the two weeks “merge window” for kernel version  $N$  begins
- ③ During the merge window Linus takes patches for kernel version  $N$  from subsystem maintainers
- ④ The merge window ends when Linus releases `-rc1` (first release candidate) of kernel version  $N$
- ⑤ Every week Linus releases another `-rc` kernel
  - Only bug fixes are accepted at this stage
- ⑥ Somewhere between `-rc6` and `-rc9`, Linus releases kernel version  $N$
- ⑦ Kernel version  $N$  moves to the `-stable` team, and receives bug fixes until a little after the release of kernel version  $N+1$
- ⑧ Some special kernel versions are maintained longer by interested parties under `-longterm`

# Overview of the Patch Acceptance Process

- ❶ Develop a feature or fix a bug, and test
- ❷ Send a patch to the maintainer(s), and Cc the mailing list
- ❸ Listen to comments, fix, and resend the patch
  - If you believe a suggested change is wrong, explain why
- ❹ Repeat the last step as necessary
- ❺ Wait for the subsystem maintainer to apply your patch
- ❻ Be responsive to problem reports regarding your patch, and fix them
- ❼ If all goes well, Linus pulls the subsystem maintainer patches during the merge window, including yours; your patch is now in the “mainline kernel”
- ❽ Sometimes long term maintenance of your code is necessary
  - Neglecting to maintain your code may lead to its removal in the long run, if nobody else shows interest

Tk Open Systems Ltd.

Solutions for an Open World



- Contributed code license must be compatible with the GNU General Public License version 2 (GPL v2)
- If you write the code as part of your job (contracted or hired), your employer must be aware of the licensing requirement
  - This is of concern mainly in large and bureaucratic companies
- No copyright assignment is required
- Submitted patches must include a Signed-off-by: tag, which bears legal significance
  - See the “Developer’s Certificate of Origin” in Documentation/SubmittingPatches for details

Disclaimer: I am not a lawyer. If in doubt, consult your local legal advisor.

# Which Kernel Version to Use as Development Base

Sometimes your work depends on features that are not present yet in released kernels. In this case select your base development kernel in the following descending order of precedence:

- ❶ The latest released kernel
- ❷ The last `-rc` development version
- ❸ The development tree of the relevant subsystem

Do not base you work on the `-next` tree.

# Finding Patch Contacts

Locate the relevant subsystem entry in the kernel MAINTAINERS file, and get the following:

- Maintainer(s)
- Mailing list
- Development source tree (git, quilt, etc.)
- Patchwork patch tracking website
- Website

## Example MAINTAINERS entry

```
LINUX FOR POWERPC (32-BIT AND 64-BIT)
M:   Benjamin Herrenschmidt <benh@kernel.crashing.org>
M:   Paul Mackerras <paulus@samba.org>
W:   http://www.penguinppc.org/
L:   linuxppc-dev@lists.ozlabs.org
Q:   http://patchwork.ozlabs.org/project/linuxppc-dev/list/
T:   git git://git.kernel.org/pub/scm/linux/kernel/git/benh/powerpc.git
S:   Supported
F:   Documentation/powerpc/
F:   arch/powerpc/
```





# Things to Do Before Sending a Patch

- Make sure that your code matches the standard kernel coding style as described in `Documentation/CodingStyle`
- Write a short and clear description of the patch, and the reason this patch is needed
  - This description becomes part of the permanent kernel git log
  - For fixes to an already released kernel, add the `"Cc: stable@kernel.org"` tag
  - See examples below
- Follow `Documentation/SubmitChecklist`
- Test your code based on a new enough kernel
- Rebase on newer kernels if they introduce relevant changes
- Monitor the subsystem mailing list for patches that are relevant to your work

# How to Generate and Send a Patch Using git

- ① When committing use the `-s` option of `git commit` to add your Signed-off-by to the commit log
- ② Generate a patch with `git format-patch`

## Example of single patch generation with git

```
$ git format-patch -o /tmp HEAD~
```

- ③ Run `scripts/checkpatch.pl` on your patch
- ④ In a revised patch add meta changelog to the generated .patch file(s)
  - This goes below the '---' line
- ⑤ Send the patch to relevant lists and maintainers

## Example of single patch sending with git

```
$ git send-email \  
  --to 'Benjamin Herrenschmidt <benh@kernel.crashing.org>' \  
  --cc linuxppc-dev@lists.ozlabs.org \  
  /tmp/powerpc-fix-something.patch
```



World

# Simple Patch Example

From: Baruch Siach <baruch@tkos.co.il>  
To: Guennadi Liakhovetski <g.liakhovetski@gmx.de>  
Cc: linux-media@vger.kernel.org, Baruch Siach <baruch@tkos.co.il>  
Subject: [PATCH] v4l: soc\_camera: fix bound checking of mbus\_fmt[] index

When code <= V4L2\_MBUS\_FMT\_FIXED soc\_mbus\_get\_fmtdesc returns a pointer to mbus\_fmt[x], where x < 0. Fix this.

Signed-off-by: Baruch Siach <baruch@tkos.co.il>

---

```
drivers/media/video/soc_mediabus.c |    2 ++
1 files changed, 2 insertions(+), 0 deletions(-)
```

diff --git a/drivers/media/video/soc\_mediabus.c b/drivers/media/video/soc\_mediabus.c  
index f8d5c87..a2808e2 100644

```
--- a/drivers/media/video/soc_mediabus.c
+++ b/drivers/media/video/soc_mediabus.c
@@ -136,6 +136,8 @@ const struct soc_mbus_pixelfmt *soc_mbus_get_fmtdesc(
 {
     if ((unsigned int)(code - V4L2_MBUS_FMT_FIXED) > ARRAY_SIZE(mbus_fmt))
         return NULL;
+    if ((unsigned int)code <= V4L2_MBUS_FMT_FIXED)
+        return NULL;
     return mbus_fmt + code - V4L2_MBUS_FMT_FIXED - 1;
 }
EXPORT_SYMBOL(soc_mbus_get_fmtdesc);
```

# Patch with Version, Changelog, and Review Tag

From: Baruch Siach <baruch@tkos.co.il>  
To: linux-kernel@vger.kernel.org  
Cc: Andrew Morton <akpm@linux-foundation.org>, Indan Zupancic <indan@nul.nu>,  
Greg KH <greg@kroah.com>, "H. Peter Anvin" <hpa@zytor.com>,  
Alex Gershgorin <agersh@rambler.ru>, Baruch Siach <baruch@tkos.co.il>  
Subject: [PATCHv3] drivers/misc: Altera active serial implementation

From: Alex Gershgorin <agersh@rambler.ru>

...

Reviewed-by: Indan Zupancic <indan@nul.nu>  
Signed-off-by: Alex Gershgorin <agersh@rambler.ru>  
Signed-off-by: Baruch Siach <baruch@tkos.co.il>  
---

Changes in v3:

- \* Rename to altera\_as for a better description of the driver scope
- \* Mention ESPC devices in the Kconfig help text
- \* Add a comment that explains why the static altera\_as\_devs arrays doesn't need locking protection
- \* Shorten too long delays
- \* Move the erase operation to a separate function
- \* Eliminate page\_count in .write, use \*ppos instead

Changes in v2:

...

# Patch Series

- Split large changes into a series of smaller logical changes
  - Easier for reviewers and maintainers
  - Separate patches for different subsystems
- The series must be “bisectable”; no single patch is allowed to break the kernel build
- Each patch in the series should be minimal
  - No need to reflect you own development history in the series
  - Don't add something in one patch only to remove it in a later one
  - With git, use interactive rebase (`git rebase -i`) to edit earlier patches in a series
- A series longer than 2 patches should include a cover letter
- When emailing, the whole series should be in a single thread
  - All emails (except the first) include the In-Reply-To header pointing to the first, which is the cover letter

# How to Generate and Send a Patch Series Using git

- ❶ Put your Signed-off-by tag in each patch
- ❷ Generate the patch series with `git format-patch`

## Example of patch series generation with git

```
$ git format-patch -o /tmp/myseries --cover-letter HEAD~5
```

- ❸ Edit the cover letter (the file with the 0000 prefix)
  - Replace the SUBJECT HERE stub subject with something sensible
  - Replace the BLURB HERE stub body text with an overall description of your patch series
  - Add series change log when applicable
- ❹ Send the patch series to relevant list and maintainers

## Example of patch series sending with git

```
$ git send-email \  
  --to 'Benjamin Herrenschmidt <benh@kernel.crashing.org>' \  
  --cc linuxppc-dev@lists.ozlabs.org \  
  /tmp/myseries/*.patch
```



World

# Patch Series Cover Letter Example

From: Baruch Siach <baruch@tkos.co.il>  
To: Sascha Hauer <kernel@pengutronix.de>  
Subject: [PATCH 0/4] mx25: add support for FEC on i.MX25 PDK  
Cc: netdev@vger.kernel.org, Baruch Siach <baruch@tkos.co.il>,  
linux-arm-kernel@lists.infradead.org

This patch series adds support for the FEC peripheral of the i.MX25 on the i.MX25 PDK board.

The first two patches are fixes for compilation and run failures. The third patch enables RMII if the FEC driver. Finally, the last patch adds the necessary board support code (pads, clock, etc.)

The FEC fix seems like an ugly hack to me. Suggestions for a better solution are welcome.

Baruch Siach (4):

- mx25: s/NO\_PAD\_CTL/NO\_PAD\_CTRL/
- mx25: don't force input on FEC pins
- fec: add support for Freescale i.MX25 PDK (3DS)
- mx25: add support for FEC on i.MX25 PDK

arch/arm/mach-mx25/clock.c		2 +
arch/arm/mach-mx25/devices.c		19 +++++++
arch/arm/mach-mx25/devices.h		1 +
arch/arm/mach-mx25/mx25pdk.c		40 ++++++
arch/arm/plat-mxc/include/mach/iomux-mx25.h		64 ++++++-----
arch/arm/plat-mxc/include/mach/mx25.h		4 ++
drivers/net/fec.c		22 +++++++
drivers/net/fec.h		2 +

8 files changed, 121 insertions(+), 33 deletions(-)

# The Review Process

- Good patches ease the work of reviewers
- Pay attention to comments, and reply to the point
- Sooner or later you are likely to see insulting language flying at your direction; don't take it personally
- Fix what you're asked to fix, or else explain why this is not needed
- Document changes to your patch in the subsequent submissions
- Wait a few days before sending another round of patches



# What to do When You get No Response

- Wait
- Update (rebase) and repost your patch as necessary
- Send polite ping messages
- Monitor the mailing list, participate in related discussions, and mention your patch
- Cc Andrew Morton if the maintainer is not responsive
- Cc the linux-kernel mailing list

# After Your Patch has been Merged

There are several exposure levels of a merged patch in ascending order:

- The subsystems maintainers' tree, and the -next tree
- Linus' tree
- A released kernel

At each level you should:

- Respond to reports of build failures and bugs
- Send fixes to reported bugs promptly, especially regressions
  - Don't forget to add "Cc: stable@kernel.org" as necessary
- Respond to patches with bug fixes or improvement suggestions
- Participate in the review of patches related to yours

# Further Info (1)

In-kernel documentation:

- `Documentation/HOWTO`
- “A Guide to the Kernel Development Process” at `Documentation/development-process/*`
- `Documentation/SubmittingPatches`
- `Documentation/SubmitChecklist`
- `Documentation/SubmittingDrivers`
- `Documentation/CodingStyle`
- `Documentation/stable_api_nonsense.txt`

## Further Info (2)

### Recommended reading:

- Dan J. Williams, *Avoiding the OS abstraction trap*  
(<http://lwn.net/Articles/454716/>)
- Jonathan Corbet, *On multi-platform drivers*  
(<http://lwn.net/Articles/457674/>)
- Jonathan Corbet, *The platform problem*  
(<http://lwn.net/Articles/443531/>)
- Andi Kleen , *On submitting kernel patches*  
(<http://halobates.de/on-submitting-patches.pdf>)

### Guides:

- <http://kernelnewbies.org/UpstreamMerge>

### Talks (video):

- Jonathan Corbet: How kernel development goes wrong and why you should be a part of it anyway (FOSDEM 2011)

<http://www.youtube.com/watch?v=MzCIBZONf5M>

