How to help improve kernel documentation

Documentation is an important part of any software-development project. Good documentation helps to bring new developers in and helps established developers work more effectively. Without top-quality documentation, a lot of time is wasted in reverse-engineering the code and making avoidable mistakes.

Unfortunately, the kernel's documentation currently falls far short of what it needs to be to support a project of this size and importance.

This guide is for contributors who would like to improve that situation. Kernel documentation improvements can be made by developers at a variety of skill levels; they are a relatively easy way to learn the kernel process in general and find a place in the community. The bulk of what follows is the documentation maintainer's list of tasks that most urgently need to be done.

The documentation TODO list

There is an endless list of tasks that need to be carried out to get our documentation to where it should be. This list contains a number of important items, but is far from exhaustive; if you see a different way to improve the documentation, please do not hold back!

Addressing warnings

The documentation build currently spews out an unbelievable number of warnings. When you have that many, you might as well have none at all; people ignore them, and they will never notice when their work adds new ones. For this reason, eliminating warnings is one of the highest-priority tasks on the documentation TODO list. The task itself is reasonably straightforward, but it must be approached in the right way to be successful.

Warnings issued by a compiler for C code can often be dismissed as false positives, leading to patches aimed at simply shutting the compiler up. Warnings from the documentation build almost always point at a real problem; making those warnings go away requires understanding the problem and fixing it at its source. For this reason, patches fixing documentation warnings should probably not say "fix a warning" in the changelog title; they should indicate the real problem that has been fixed.

Another important point is that documentation warnings are often created by problems in kerneldoc comments in C code. While the documentation maintainer appreciates being copied on fixes for these warnings, the documentation tree is often not the right one to actually carry those fixes; they should go to the maintainer of the subsystem in question.

For example, in a documentation build I grabbed a pair of warnings nearly at random:

```
./drivers/devfreq/devfreq.c:1818: warning: bad line:
      - Resource-managed devfreq_register_notifier()
./drivers/devfreq/devfreq.c:1854: warning: bad line:
      - Resource-managed devfreq_unregister_notifier()
```

(The lines were split for readability).

A quick look at the source file named above turned up a couple of kerneldoc comments that look like this:

```
/**
 * devm_devfreq_register_notifier()
        - Resource-managed devfreq_register_notifier()
 * @dev:      The devfreq user device. (parent of devfreq)
 * @devfreq:  The devfreq object.
 * @nb:               The notifier block to be unregistered.
 * @list:     DEVFREQ_TRANSITION_NOTIFIER.
 */
```

The problem is the missing "*", which confuses the build system's simplistic idea of what C comment blocks look like. This problem had been present since that comment was added in 2016 â€” a full four years. Fixing it was a matter of adding the missing asterisks. A quick look at the history for that file showed what the normal format for subject lines is, and `scripts/get_maintainer.pl` told me who should receive it. The resulting patch looked like this:

```
[PATCH] PM / devfreq: Fix two malformed kerneldoc comments

Two kerneldoc comments in devfreq.c fail to adhere to the required format,
resulting in these doc-build warnings:

  ./drivers/devfreq/devfreq.c:1818: warning: bad line:
        - Resource-managed devfreq_register_notifier()
  ./drivers/devfreq/devfreq.c:1854: warning: bad line:
        - Resource-managed devfreq_unregister_notifier()

Add a couple of missing asterisks and make kerneldoc a little happier.

Signed-off-by: Jonathan Corbet <corbet@lwn.net>
---
 drivers/devfreq/devfreq.c | 4 ++--
 1 file changed, 2 insertions(+), 2 deletions(-)

diff --git a/drivers/devfreq/devfreq.c b/drivers/devfreq/devfreq.c
index 57f6944d65a6..00c9b80b3d33 100644
--- a/drivers/devfreq/devfreq.c
+++ b/drivers/devfreq/devfreq.c
@@ -1814,7 +1814,7 @@ static void devm_devfreq_notifier_release(struct device *dev, void *res)

 /**
  * devm_devfreq_register_notifier()
-     - Resource-managed devfreq_register_notifier()
+ *   - Resource-managed devfreq_register_notifier()
  * @dev:     The devfreq user device. (parent of devfreq)
  * @devfreq: The devfreq object.
  * @nb:              The notifier block to be unregistered.
@@ -1850,7 +1850,7 @@ EXPORT_SYMBOL(devm_devfreq_register_notifier);

 /**
  * devm_devfreq_unregister_notifier()
-     - Resource-managed devfreq_unregister_notifier()
+ *   - Resource-managed devfreq_unregister_notifier()
  * @dev:     The devfreq user device. (parent of devfreq)
  * @devfreq: The devfreq object.
  * @nb:              The notifier block to be unregistered.
--
2.24.1
```

The entire process only took a few minutes. Of course, I then found that somebody else had fixed it in a separate tree, highlighting another lesson: always check linux-next to see if a problem has been fixed before you dig into it.

Other fixes will take longer, especially those relating to structure members or function parameters that lack documentation. In such cases, it is necessary to work out what the role of those members or parameters is and describe them correctly. Overall, this task gets a little tedious at times, but it's highly important. If we can actually eliminate warnings from the documentation build, then we can start expecting developers to avoid adding new ones.

Languishing kerneldoc comments

Developers are encouraged to write kerneldoc comments for their code, but many of those comments are never pulled into the docs build. That makes this information harder to find and, for example, makes Sphinx unable to generate links to that documentation. Adding `kernel-doc` directives to the documentation to bring those comments in can help the community derive the full value of the work that has gone into creating them.

The `scripts/find-unused-docs.sh` tool can be used to find these overlooked comments.

Note that the most value comes from pulling in the documentation for exported functions and data structures. Many subsystems also have kerneldoc comments for internal use; those should not be pulled into the documentation build unless they are placed in a document that is specifically aimed at developers working within the relevant subsystem.

Typo fixes

Fixing typographical or formatting errors in the documentation is a quick way to figure out how to create and send patches, and it is a useful service. I am always willing to accept such patches. That said, once you have fixed a few, please consider moving on to more advanced tasks, leaving some typos for the next beginner to address.

Please note that some things are *not* typos and should not be "fixed":

- Both American and British English spellings are allowed within the kernel documentation. There is no need to fix one by replacing it with the other.
- The question of whether a period should be followed by one or two spaces is not to be debated in the context of kernel documentation. Other areas of rational disagreement, such as the "Oxford comma", are also off-topic here.

As with any patch to any project, please consider whether your change is really making things better.

Ancient documentation

Some kernel documentation is current, maintained, and useful. Some documentation is ... not. Dusty, old, and inaccurate documentation can mislead readers and casts doubt on our documentation as a whole. Anything that can be done to address such problems is more than welcome.

Whenever you are working with a document, please consider whether it is current, whether it needs updating, or whether it should perhaps be removed altogether. There are a number of warning signs that you can pay attention to here:

- References to 2.x kernels
- Pointers to SourceForge repositories
- Nothing but typo fixes in the history for several years

- Discussion of pre-Git workflows

The best thing to do, of course, would be to bring the documentation current, adding whatever information is needed. Such work often requires the cooperation of developers familiar with the subsystem in question, of course. Developers are often more than willing to cooperate with people working to improve the documentation when asked nicely, and when their answers are listened to and acted upon.

Some documentation is beyond hope; we occasionally find documents that refer to code that was removed from the kernel long ago, for example. There is surprising resistance to removing obsolete documentation, but we should do that anyway. Extra cruft in our documentation helps nobody.

In cases where there is perhaps some useful information in a badly outdated document, and you are unable to update it, the best thing to do may be to add a warning at the beginning. The following text is recommended:

```
.. warning ::
      This document is outdated and in need of attention.  Please use
      this information with caution, and please consider sending patches
      to update it.
```

That way, at least our long-suffering readers have been warned that the document may lead them astray.

Documentation coherency

The old-timers around here will remember the Linux books that showed up on the shelves in the 1990s. They were simply collections of documentation files scrounged from various locations on the net. The books have (mostly) improved since then, but the kernel's documentation is still mostly built on that model. It is thousands of files, almost each of which was written in isolation from all of the others. We don't have a coherent body of kernel documentation; we have thousands of individual documents.

We have been trying to improve the situation through the creation of a set of "books" that group documentation for specific readers. These include:

- Documentation/admin-guide/index.rst
- Documentation/core-api/index.rst
- Documentation/driver-api/index.rst
- Documentation/userspace-api/index.rst

As well as this book on documentation itself.

Moving documents into the appropriate books is an important task and needs to continue. There are a couple of challenges associated with this work, though. Moving documentation files creates short-term pain for the people who work with those files; they are understandably unenthusiastic about such changes. Usually the case can be made to move a document once; we really don't want to keep shifting them around, though.

Even when all documents are in the right place, though, we have only managed to turn a big pile into a group of smaller piles. The work of trying to knit all of those documents together into a single whole has not yet begun. If you have bright ideas on how we could proceed on that front, we would be more than happy to hear them.

Stylesheet improvements

With the adoption of Sphinx we have much nicer-looking HTML output than we once did. But it could still use a lot of improvement; Donald Knuth and Edward Tufte would be unimpressed. That requires tweaking our stylesheets to create more typographically sound, accessible, and readable output.

Be warned: if you take on this task you are heading into classic bikeshed territory. Expect a lot of opinions and discussion for even relatively obvious changes. That is, alas, the nature of the world we live in.

Non-LaTeX PDF build

This is a decidedly nontrivial task for somebody with a lot of time and Python skills. The Sphinx toolchain is relatively small and well contained; it is easy to add to a development system. But building PDF or EPUB output requires installing LaTeX, which is anything but small or well contained. That would be a nice thing to eliminate.

The original hope had been to use the rst2pdf tool (https://rst2pdf.org/) for PDF generation, but it turned out to not be up to the task. Development work on rst2pdf seems to have picked up again in recent times, though, which is a hopeful sign. If a suitably motivated developer were to work with that project to make rst2pdf work with the kernel documentation build, the world would be eternally grateful.

Write more documentation

Naturally, there are massive parts of the kernel that are severely underdocumented. If you have the knowledge to document a specific kernel subsystem and the desire to do so, please do not hesitate to do some writing and contribute the result to the kernel. Untold numbers of kernel developers and users will thank you.