MAIN TAIN ERS truth and ficti on

[LWN subscri ber-only content]

By **Jonathan Corbet**
January 14, 2021

Since the release of the 5.5 kernel in January 2020, there have been almost 87,000 patches from just short of 4,600 developers merged into the mainline repository. Reviewing all of those patches would be a tall order for even the most prolific of kernel developers, so decisions on patch acceptance are delegated to a long list of subsystem maintainers, each of whom takes partial or full responsibility for a specific portion of the kernel. These maintainers are documented in a file called, surprisingly, `MAINTAINERS`. But the `MAINTAINERS` file, too, must be maintained; how well does it reflect reality?

The `MAINTAINERS` file doesn't exist just to give credit to maintainers; developers make use of it to know where to send patches. The `get_maintainer.pl` script automates this process by looking at the files modified by a patch and generating a list of email addresses to send it to. Given that misinformation in this file can send patches astray, one would expect it to be kept up-to-date. Recently, your editor received a suggestion from Jakub Kicinski that there may be insights to be gleaned from comparing `MAINTAINERS` entries against activity in the real world. A bit of Python bashing later, a new analysis script was born.

Di ggi ng i MAINTAINERS

There are, it turns out, 2,280 "subsystems" listed in the `MAINTAINERS` file. Each of those subsystems includes a list of covered files and directories. One can look at the commits applied against those files to see who has been working in any given subsystem; writing patches obviously qualifies as this sort of work, but so do other activities like handling patches (as indicated by `Signed-off-by` tags) or reviewing them (`Reviewed-by` or `Acked-by`). By making use of a bit of CPU time diverted from cryptocurrency mining, it is possible to come up with an approximation of when a given subsystem's listed maintainers last actually did some work in that subsystem.

The full results of this analysis are available for those wanting to see the details.

There are, however, ways of narrowing down the data a bit to pick out some of the more interesting artifacts in this file. For example, there are 367 subsystems for which there is no maintainer or the maintainer has never been seen in the entire Git history (excluding "subsystems" with no files — see below). In many of these cases, the subsystem itself is well past the prime of its life; there simply isn't a lot of work for a 3c59x network-card maintainer to do these days. The networking developers are not buried in ATM patches, the Palm Treo hasn't seen much support work, Apple has released few M68k systems recently, there aren't many Arm floppy drives still in use, and S3 Savage video cards just aren't the must-have device they once were. Many of these entries are likely to point to code that could be removed altogether.

Similar lessons can be drawn from the list of subsystems with no listed maintainers at all. Of course, some of those are rather vague in other ways as well; one subsystem is simply called "ABI/API" and points to the `inux-api` mailing list. There is actually one file associated with this "subsystem"; it's `kernel/sys_ni.c`, which handles calls to non-implemented system calls. This entry is thus an attempt to get developers to copy the `inux-api` list when they add new system calls. A similar entry exists for "Arm subarchitectures".

Some maintainerless subsystems, such as the framebuffer layer, could probably benefit from somebody willing to take them over. The reiserfs filesystem lacks a maintainer but still seems to have some users.

Others, like DECnet or the Matrox framebuffer, are probably best left alone (or removed) at this point.

Some "subsystems" listed in the `MAINTAINERS` file have no files to maintain; one interesting example is "embedded Linux", allegedly maintained by Paul Gortmaker, Matt Mackall, and David Woodhouse. Given the success of embedded Linux, one can only assume that they are doing an outstanding job. The "device number registry" claims to be maintained, but the entry contains only a pointer to a nonexistent web page. The URLs in the "disk geometry and partition handling" entry still work, but the pages do not appear to have been updated for well over a decade; not much is happening with Zip drive geometry these days, it would appear. The man pages, instead, are actively maintained, but they do not exist within the kernel tree.

Help needed

There are a couple of conclusions that can be drawn from the results so far. One is that many kernel subsystems are not really in need of maintenance at this point; some of them, instead, may be in need of removal. Another is that perhaps the `MAINTAINERS` file itself is in need of a bit of cleanup in spots. But it is also worth asking whether this data can be used to spot subsystems that could benefit from a new maintainer. To answer that question, some additional CPU time was expended to find all subsystems meeting these criteria:

- There is either no listed maintainer or the alleged maintainers have been inactive in that subsystem for at least six months.
- At least 50 commits have touched that subsystem since the release of the 5.5 kernel in January 2020.

The idea behind this search was to find subsystems that are still undergoing some sort of active development, but which do not have an active, listed maintainer. The results can be divided into a few different categories.

Some `MAINTAINERS` entries have broad lists of covered files that make the commit count seem larger than it really is. For example, the subsystem named "ASYNCHRONOUS TRANSFERS/TRANSFORMS (IOAT) API" includes all of `drivers/dma`, which is also claimed by "DMA GENERIC OFFLOAD ENGINE SUBSYSTEM". That subsystem, in turn, is actively maintained by Vinod Koul. There are two subsystems that fall into this category; in the tables below "Activity" indicates the last observed activity by the listed maintainers (if any), while "Commits" shows the number of commits affecting the subsystem since 5.5:

Subsystem	Activity	Commits
ASYNCHRONOUS TRANSFERS/TRANSFORMS (IOAT) API	——	536
HISILICON NETWORK SUBSYSTEM DRIVER	2019-11-16	258

These subsystems either do not exist as a separate entity, or they should have their lists of covered files reduced to match reality.

Then, there are the subsystems where the maintainers hide behind a corporate email alias. The listed maintainer for "DIALOG SEMICONDUCTOR DRIVERS" is `support.opensource@diasemi.com`, which is obviously not an address that will appear in any actual commits. A look within that subsystem shows active reviews from diasemi.com addresses, though, so the subsystem cannot really be said to be unmaintained. This category contains:

Subsystem	Activity	Commits
DIALOG SEMICONDUCTOR DRIVERS	——	120
QUALCOMM ATHEROS ATH9K WIRELESS DRIVER	——	65
WOLFSON MICROELECTRONICS DRIVERS	——	146

Related to the above are subsystems where the maintainer entry is simply out of date; the listed maintainer is inactive, but somebody else, often from the same company, has picked up the slack and is acting as a de-facto

maintainer. These include:

Subsystem	Activity	Commits
HISILICON NETWORK SUBSYSTEM 3 DRIVER (HNS3)	2019-11-16	234
HISILICON SECURITY ENGINE V2 DRIVER (SEC2)	2020-06-18	55
LINUX FOR POWER MACINTOSH	2018-10-19	71
MELLANOX ETHERNET INNOVA DRIVERS	——	93
MELLANOX MLX4 IB driver	——	70
OMAP HWMOD DATA	2016-06-10	102
QUALCOM AUDIO (ASoC) DRIVERS	2018-05-21	125
TEGRA I2C DRIVER	2018-05-30	56

Finally, there are the subsystems that truly seem to lack a maintainer; they typically show patterns of commits either merged by a variety of subsystem maintainers, or passing through one of a few maintainers of last resort. They are:

Subsystem	Activity	Commits
ARM/UNIPHIER ARCHITECTURE	——	73
DRBD DRIVER	2018-12-20	51
FRAMEBUFFER LAYER	——	402
HMM - Heterogeneous Memory Management	2020-05-19	54
I2C SUBSYSTEM HOST DRIVERS	——	434
MARVELL MVNETA ETHERNET DRIVER	2018-11-23	65
MEDIA DRIVERS FOR RENESAS - VIN	2019-10-10	56
MUSB MULTIPOINT HIGH SPEED DUAL-ROLE CONTROLLER	2020-06-24	54
NFC SUBSYSTEM	——	72
PROC FILESYSTEM	——	171
PROC SYSCTL	2020-06-08	51
QLOGIC QLGE 10Gb ETHERNET DRIVER	2019-10-04	77
STAGING - REALTEK RTL8188EU DRIVERS	2020-07-15	121
STMMAC ETHERNET DRIVER	2020-05-01	174
UNIVERSAL FLASH STORAGE HOST CONTROLLER DRIVER	——	277
USB NETWORKING DRIVERS	——	119
X86 PLATFORM DRIVERS - ARCH	——	120

Most of the above will be unsurprising to people who have been paying attention to the areas in question. The framebuffer subsystem is a known problem area; the "soft scrollback" capability was recently removed from the framebuffer driver due to a lack of maintainership. Quite a few people depend on this code still, but it is increasingly difficult to integrate with the kernel's graphics drivers and few people have any appetite to delve into it.

The I2C host drivers do, in fact, have a de-facto maintainer; it's Wolfram Song, who also maintains the core I2C subsystem. He has long wished for help maintaining those drivers but none seems to be forthcoming, so he takes care of them in the time that is available. `/proc` is an interesting example; everybody depends on it, but nobody has taken responsibility for its maintenance. HMM, too, is interesting; its creator went to a lot of effort to get the code merged, but appears to have moved on to other pursuits now.

All of the above look like places where aspiring kernel developers could lend a welcome hand.

What about subsystems that have no entry in the `MAINTAINERS` file at all? If one were to bash out a quick script to find all files in the kernel tree that are not covered by at least one line in `MAINTAINERS`, one would end up with a list of just over 2,800 files. These include the `MAINTAINERS` file itself, naturally. Of the rest, the vast majority are header files under `include/`, most of which probably do have maintainers and should be added to the appropriate entries. Discouragingly, there are 72 files under `kernel/` without a listed maintainer — a situation which certainly does not reflect reality. The SYSV IPC code is unmaintained, reflecting its generally unloved nature. Most of the rest of the unmaintained files are under `tools/` or `samples/`.

A harder case to find is that of files that are covered by a `MAINTAINERS` entry, but which are not actually maintained by the named person; this will happen often with entries that cover entire directory trees. Your editor is listed as handling all of `Documentation`, but certainly cannot be said to be "maintaining" many of those files, for example; this is a situation that will arise in many places in the kernel tree.

If one were to try to draw some overall conclusions from this data, they might read something like the following. The `MAINTAINERS` file definitely has some dark corners that could, themselves, use some maintenance (some of which is already being done). There are some parts of the kernel lacking maintainers that could definitely use one, and other parts that have aged beyond the point of needing maintenance. For the most part, though, the subsystems in the kernel have designated maintainers, and most of them are at least trying to take care of the code they have responsibility for. The situation could be a lot worse.

[As usual, the script used to generate the above tables can be found in the gitdm repository at `git://git.lwn.net/gitdm.git`.]

Index entries for this article
Kernel Development model/Maintainers

Send a free link
Post a comment